M68HC16 R SERIES

(MC68HC16R1, MC68HC916R1, MC68HC16R3, MC68HC916R3) USER'S MANUAL

PRELIMINARY

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters can and do vary in different applications. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part.
MOTOROLA and ! are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

© MOTOROLA, INC. 1998

TABLE OF CONTENTS

Paragraph **Title** **Page**

SECTION 1
INTRODUCTION

SECTION 2
NOMENCLATURE

2.1 Symbols and Operators 2-1
2.2 CPU16 Register Mnemonics 2-2
2.3 Register Mnemonics 2-3
2.4 Conventions 2-6

SECTION 3
OVERVIEW

3.1 M68HC16 R-Series MCU Features 3-1
3.1.1 Central Processor Unit (CPU16) 3-1
3.1.2 Single-Chip Integration Module 2 (SCIM2) 3-1
3.1.3 Standby RAM (SRAM) 3-1
3.1.4 Masked ROM Module (MRM) 3-2
3.1.5 Flash EEPROM Modules (FLASH) 3-2
3.1.6 Block Erasable Flash EEPROM (BEFLASH) 3-2
3.1.7 Analog-to-Digital Converter (ADC) 3-2
3.1.8 Multichannel Communication Interface (MCCI) 3-2
3.1.9 Configurable Timer Module (CTM) 3-2
3.2 Intermodule Bus 3-3
3.3 System Block Diagram and Pin Assignment Diagrams 3-3
3.4 Pin Descriptions 3-12
3.5 CPU16 Memory Mapping 3-19
3.6 Internal Register Maps 3-21
3.7 Address Space Maps 3-25

SECTION 4
CENTRAL PROCESSOR UNIT

4.1 General 4-1
4.2 Register Model 4-1
4.2.1 Accumulators 4-3
4.2.2 Index Registers 4-3
4.2.3 Stack Pointer 4-3
4.2.4 Program Counter 4-3
4.2.5 Condition Code Register 4-4
4.2.6 Address Extension Register and Address Extension Fields 4-5

TABLE OF CONTENTS
(Continued)

Paragraph **Title** **Page**

4.2.7 Multiply and Accumulate Registers 4-5
4.3 Memory Management 4-5
4.3.1 Address Extension 4-6
4.3.2 Extension Fields 4-6
4.4 Data Types 4-6
4.5 Memory Organization 4-7
4.6 Addressing Modes 4-8
4.6.1 Immediate Addressing Modes 4-9
4.6.2 Extended Addressing Modes 4-10
4.6.3 Indexed Addressing Modes 4-10
4.6.4 Inherent Addressing Mode 4-10
4.6.5 Accumulator Offset Addressing Mode 4-10
4.6.6 Relative Addressing Modes 4-10
4.6.7 Post-Modified Index Addressing Mode 4-10
4.6.8 Use of CPU16 Indexed Mode to Replace M68HC11 Direct Mode 4-11
4.7 Instruction Set 4-11
4.7.1 Instruction Set Summary 4-11
4.8 Comparison of CPU16 and M68HC11 CPU Instruction Sets 4-31
4.9 Instruction Format 4-33
4.10 Execution Model 4-34
4.10.1 Microsequencer 4-35
4.10.2 Instruction Pipeline 4-35
4.10.3 Execution Unit 4-35
4.11 Execution Process 4-36
4.11.1 Changes in Program Flow 4-36
4.12 Instruction Timing 4-36
4.13 Exceptions 4-37
4.13.1 Exception Vectors 4-37
4.13.2 Exception Stack Frame 4-38
4.13.3 Exception Processing Sequence 4-39
4.13.4 Types of Exceptions 4-39
4.13.4.1 Asynchronous Exceptions 4-39
4.13.4.2 Synchronous Exceptions 4-39
4.13.5 Multiple Exceptions 4-40
4.13.6 RTI Instruction 4-40
4.14 Development Support 4-40
4.14.1 Deterministic Opcode Tracking 4-40
4.14.1.1 IPIPE0/IPIPE1 Multiplexing 4-41
4.14.1.2 Combining Opcode Tracking with Other Capabilities 4-41
4.14.2 Breakpoints 4-41
4.14.3 Opcode Tracking and Breakpoints 4-42

TABLE OF CONTENTS
(Continued)

Paragraph **Title** **Page**

4.14.4 Background Debug Mode 4-42
4.14.5 Enabling BDM 4-42
4.14.5.1 BDM Sources 4-42
4.14.5.2 Entering BDM 4-43
4.14.5.3 BDM Commands 4-43
4.14.5.4 Returning from BDM 4-44
4.14.5.5 BDM Serial Interface 4-44
4.15 Recommended BDM Connection 4-45
4.16 Digital Signal Processing 4-46

SECTION 5
SINGLE-CHIP INTEGRATION MODULE 2

5.1 General 5-1
5.2 System Configuration 5-2
5.2.1 Module Mapping 5-2
5.2.2 Interrupt Arbitration 5-3
5.2.3 Noise Reduction in Single-Chip Mode 5-3
5.2.4 Show Internal Cycles 5-4
5.2.5 FREEZE Assertion Response 5-4
5.3 System Clock 5-4
5.3.1 System Clock Sources 5-4
5.3.2 Clock Synthesizer Operation 5-5
5.3.3 Slow Reference Mode 5-6
5.3.4 Clock Speed Changes in Slow Reference Mode 5-9
5.3.5 Fast Reference Mode 5-14
5.3.6 Clock Speed Changes in Fast Reference Mode 5-17
5.3.7 External Clock Mode 5-20
5.3.8 M6800 Bus Clock 5-21
5.3.9 Low-Power Operation 5-21
5.3.10 Loss of Reference Signal 5-23
5.4 System Protection 5-24
5.4.1 Reset Status 5-25
5.4.2 Bus Monitor 5-25
5.4.3 Halt Monitor 5-25
5.4.4 Spurious Interrupt Monitor 5-25
5.4.5 Software Watchdog 5-26
5.4.6 Periodic Interrupt Timer 5-28
5.4.7 Interrupt Priority and Vectoring for the Periodic Interrupt Timer 5-29
5.4.8 Low-Power Stop Mode 5-29
5.5 External Bus Interface 5-30
5.5.1 Bus Control Signals 5-30

TABLE OF CONTENTS
(Continued)

Paragraph **Title** **Page**

5.5.1.1 Address Bus 5-30
5.5.1.2 Address Strobe 5-30
5.5.1.3 Data Bus 5-30
5.5.1.4 Data Strobe 5-32
5.5.1.5 Read/Write Signal 5-32
5.5.1.6 Size Signals 5-32
5.5.1.7 Function Codes 5-32
5.5.1.8 Data Size Acknowledge Signals 5-33
5.5.1.9 Bus Error Signal 5-33
5.5.1.10 Halt Signal 5-33
5.5.1.11 Autovector Signal 5-33
5.5.2 Dynamic Bus Sizing 5-33
5.5.3 Operand Alignment 5-35
5.5.4 Misaligned Operands 5-35
5.5.5 Operand Transfer Cases 5-35
5.6 Bus Operation 5-36
5.6.1 Synchronization to CLKOUT 5-37
5.6.2 Regular Bus Cycle 5-37
5.6.2.1 Read Cycle 5-38
5.6.2.2 Write Cycle 5-38
5.6.3 Fast Termination Cycles 5-39
5.6.4 CPU Space Cycles 5-40
5.6.4.1 Breakpoint Acknowledge Cycle 5-41
5.6.4.2 LPSTOP Broadcast Cycle 5-42
5.6.5 Alternate Bus Cycle Termination 5-43
5.6.5.1 Bus Errors 5-44
5.6.5.2 Double Bus Faults 5-45
5.6.5.3 Halt Operation 5-45
5.6.6 External Bus Arbitration 5-46
5.6.6.1 Show Cycles 5-47
5.7 Reset 5-48
5.7.1 Reset Exception Processing 5-48
5.7.2 Reset Source Summary 5-48
5.7.3 Reset Processing Summary 5-49
5.7.4 Reset Status Register 5-50
5.7.5 Reset Timing 5-50
5.7.6 Power-On Reset 5-50
5.7.7 Pin State During Reset 5-52
5.7.7.1 Reset States of SCIM2 Pins 5-52
5.7.7.2 Reset States of Pins Assigned to Other MCU Modules 5-53
5.7.8 Operating Configuration Out of Reset 5-53

TABLE OF CONTENTS
(Continued)

Paragraph **Title** **Page**

5.7.8.1 Operating Mode Selection 5-54
5.7.8.2 Data Bus Mode Selection 5-55
5.7.8.3 Single-Chip Mode 5-57
5.7.8.4 Fully (16-bit) Expanded Mode 5-59
5.7.8.5 Partially (8-bit) Expanded Mode 5-60
5.7.8.6 Breakpoint Mode Selection 5-61
5.7.8.7 Emulation Mode Selection 5-62
5.7.9 Use of the Three-State Control Pin 5-62
5.8 Interrupts 5-63
5.8.1 Interrupt Exception Processing 5-63
5.8.2 Interrupt Priority and Recognition 5-63
5.8.3 Interrupt Acknowledge and Arbitration 5-64
5.8.4 Interrupt Processing Summary 5-66
5.9 Chip-Selects 5-66
5.9.1 Chip-Select Registers 5-68
5.9.1.1 Chip-Select Pin Assignment Registers 5-69
5.9.1.2 Chip-Select Base Address Registers 5-70
5.9.1.3 Chip-Select Option Registers 5-71
5.9.1.4 Port C Data Register 5-73
5.9.2 Chip-Select Operation 5-73
5.9.3 Using Chip-Select Signals for Interrupt Acknowledge Cycle Termination 5-74
5.9.4 Chip-Select Reset Operation 5-75
5.10 General-Purpose Input/Output 5-76
5.10.1 Ports A and B 5-77
5.10.2 Port E 5-77
5.10.3 Port F 5-78
5.10.4 Port G 5-80
5.10.5 Port H 5-80
5.11 Factory Test 5-81

SECTION 6
STANDBY RAM MODULE

6.1 SRAM Register Block 6-1
6.2 SRAM Array Address Mapping 6-1
6.3 SRAM Array Address Space Type 6-2
6.4 Standby Power Loss Detection 6-2
6.5 Normal Access 6-2
6.6 Standby and Low-Power Stop Operation 6-2
6.7 Reset 6-3

TABLE OF CONTENTS
(Continued)

Paragraph **Title** **Page**

SECTION 7
MASKED ROM MODULE

7.1 MRM Register Block 7-1
7.2 MRM Array Address Mapping 7-1
7.3 MRM Array Address Space Type 7-2
7.4 Normal Access 7-2
7.5 Low-Power Stop Mode Operation 7-3
7.6 ROM Signature 7-3
7.7 Reset 7-3

SECTION 8
FLASH EEPROM MODULE

8.1 Flash EEPROM Register Block 8-1
8.2 Flash EEPROM Array Address Mapping 8-2
8.3 Flash EEPROM Array Address Space Type 8-2
8.4 Normal Access 8-2
8.5 Low-Power Stop Mode Operation 8-3
8.6 Reset 8-3
8.7 Program/Erase Operation 8-3
8.7.1 Programming Sequence 8-4
8.7.2 Erasure Sequence 8-6
8.7.3 Program/Erase Voltage Signal Conditioning 8-8

SECTION 9
BLOCK-ERASABLE FLASH EEPROM

9.1 Block-Erasable Flash EEPROM Register Block 9-2
9.2 Block-Erasable Flash EEPROM Array Address Mapping 9-2
9.3 Block-Erasable Flash EEPROM Array Address Space Type 9-2
9.4 Normal Access 9-3
9.5 Low-Power Stop Mode Operation 9-3
9.6 Reset 9-3
9.7 Program/Erase Operation 9-3
9.7.1 Programming Sequence 9-4
9.7.2 Erasure Sequence 9-6
9.7.3 Program/Erase Voltage Signal Conditioning 9-9

SECTION 10
ANALOG-TO-DIGITAL CONVERTER

TABLE OF CONTENTS
(Continued)

Paragraph **Title** **Page**

10.1 General 10-1
10.2 External Connections 10-1
10.2.1 Analog Input Pins 10-2
10.2.2 Analog Reference Pins 10-3
10.2.3 Analog Supply Pins 10-3
10.3 Programmer's Model 10-3
10.4 Low-Power Stop Mode 10-3
10.5 Freeze Assertion Response 10-3
10.6 Analog Subsystem 10-4
10.6.1 Multiplexer 10-4
10.6.2 Sample Capacitor and Buffer Amplifier 10-5
10.6.3 RC DAC Array 10-5
10.6.4 Comparator 10-6
10.7 Digital Control Subsystem 10-6
10.7.1 Control/Status Registers 10-6
10.7.2 Clock and Prescaler Control 10-6
10.7.3 Sample Time 10-7
10.7.4 Resolution 10-7
10.7.5 Conversion Control Logic 10-7
10.7.6 Conversion Timing 10-12
10.7.7 Successive Approximation Register 10-13
10.7.8 Result Registers 10-13
10.8 Pin Considerations 10-14
10.8.1 Analog Reference Pins 10-14
10.8.2 Analog Power Pins 10-14
10.8.3 Analog Supply Filtering and Grounding 10-16
10.8.4 Accommodating Positive/Negative Stress Conditions 10-18
10.8.5 Analog Input Considerations 10-20
10.8.6 Analog Input Pins 10-22
10.8.6.2 Error Resulting from Leakage 10-24

SECTION 11
MULTICHANNEL COMMUNICATION INTERFACE

11.1 General 11-1
11.2 MCCI Registers and Address Map 11-2
11.2.1 MCCI Global Registers 11-2
11.2.1.1 Low-Power Stop Mode 11-2
11.2.1.2 Privilege Levels 11-3
11.2.1.3 MCCI Interrupts 11-3
11.2.2 Pin Control and General-Purpose I/O 11-4
11.3 Serial Peripheral Interface (SPI) 11-4

TABLE OF CONTENTS
(Continued)

Paragraph **Title** **Page**

11.3.1 SPI Registers 11-6
11.3.1.1 SPI Control Register (SPCR) 11-6
11.3.1.2 SPI Status Register (SPSR) 11-6
11.3.1.3 SPI Data Register (SPDR) 11-6
11.3.2 SPI Pins 11-6
11.3.3 SPI Operating Modes 11-7
11.3.3.1 Master Mode 11-7
11.3.3.2 Slave Mode 11-8
11.3.4 SPI Clock Phase and Polarity Controls 11-8
11.3.4.1 CPHA = 0 Transfer Format 11-9
11.3.4.2 CPHA = 1 Transfer Format 11-10
11.3.5 SPI Serial Clock Baud Rate 11-11
11.3.6 Wired-OR Open-Drain Outputs 11-11
11.3.7 Transfer Size and Direction 11-11
11.3.8 Write Collision 11-12
11.3.9 Mode Fault 11-12
11.4 Serial Communication Interface (SCI) 11-13
11.4.1 SCI Registers 11-13
11.4.1.1 SCI Control Registers 11-13
11.4.1.2 SCI Status Register 11-16
11.4.1.3 SCI Data Register 11-16
11.4.2 SCI Pins 11-16
11.4.3 Receive Data Pins (RXDA, RXDB) 11-17
11.4.4 Transmit Data Pins (TXDA, TXDB) 11-17
11.4.5 SCI Operation 11-17
11.4.5.1 Definition of Terms 11-17
11.4.5.2 Serial Formats 11-18
11.4.5.3 Baud Clock 11-18
11.4.5.4 Parity Checking 11-19
11.4.5.5 Transmitter Operation 11-19
11.4.5.6 Receiver Operation 11-20
11.4.5.7 Idle-Line Detection 11-21
11.4.5.8 Receiver Wake-Up 11-22
11.4.5.9 Internal Loop 11-22
11.5 MCCI Initialization 11-23

SECTION 12
CONFIGURABLE TIMER MODULE

12.1 General 12-1
12.2 Address Map 12-4
12.3 Time Base Bus System 12-4

TABLE OF CONTENTS
(Continued)

Paragraph **Title** **Page**

12.4 Bus Interface Unit Submodule (BIUSM) 12-5
12.4.1 STOP Effect On the BIUSM 12-5
12.4.2 Freeze Effect On the BIUSM 12-5
12.4.3 LPSTOP Effect on the BIUSM 12-6
12.4.4 BIUSM Registers 12-6
12.5 Counter Prescaler Submodule (CPSM) 12-6
12.5.1 CPSM Registers 12-7
12.6 Free-Running Counter Submodule (FCSM) 12-7
12.6.1 FCSM Counter 12-8
12.6.2 FCSM Clock Sources 12-8
12.6.3 FCSM External Event Counting 12-9
12.6.4 FCSM Time Base Bus Driver 12-9
12.6.5 FCSM Interrupts 12-9
12.6.6 FCSM Registers 12-9
12.7 Modulus Counter Submodule (MCSM) 12-9
12.7.1 MCSM Modulus Latch 12-10
12.7.2 MCSM Counter 12-10
12.7.2.1 Loading the MCSM Counter Register 12-11
12.7.2.2 Using the MCSM as a Free-Running Counter 12-11
12.7.3 MCSM Clock Sources 12-11
12.7.4 MCSM External Event Counting 12-11
12.7.5 MCSM Time Base Bus Driver 12-11
12.7.6 MCSM Interrupts 12-12
12.7.7 MCSM Registers 12-12
12.8 Single Action Submodule (SASM) 12-12
12.8.1 SASM Interrupts 12-13
12.8.2 SASM Registers 12-14
12.9 Double-Action Submodule (DASM) 12-14
12.9.1 DASM Interrupts 12-16
12.9.2 DASM Registers 12-16
12.10 Pulse-Width Modulation Submodule (PWMSM) 12-16
12.10.1 Output Flip-Flop and Pin 12-17
12.10.2 Clock Selection 12-17
12.10.3 PWMSM Counter 12-18
12.10.4 PWMSM Period Registers and Comparator 12-18
12.10.5 PWMSM Pulse-Width Registers and Comparator 12-19
12.10.6 PWMSM Coherency 12-19
12.10.7 PWMSM Interrupts 12-19
12.10.8 PWM Frequency 12-20
12.10.9 PWM Pulse Width 12-21
12.10.10 PWM Period and Pulse Width Register Values 12-21

TABLE OF CONTENTS
(Continued)

Paragraph **Title** **Page**

12.10.10.1 PWM Duty Cycle Boundary Cases 12-22
12.10.11 PWMSM Registers 12-22
12.11 CTM Interrupts 12-22

APPENDIX A
ELECTRICAL CHARACTERISTICS

APPENDIX B
MECHANICAL DATA AND ORDERING INFORMATION

B.1 Obtaining Updated M68HC16 R-Series MCU Mechanical Information B-7
B.2 Obtaining Ordering Information B-7

APPENDIX C
DEVELOPMENT SUPPORT

C.1 M68MMDS1632 Modular Development System C-1
C.2 M68MEVB1632 Modular Evaluation Board C-1

APPENDIX D
REGISTER SUMMARY

D.1 Central Processing Unit D-1
D.1.1 Condition Code Register D-3
D.2 Single-Chip Integration Module 2 ..... D-4
D.2.1 SCIM Configuration Register D-5
D.2.2 SCIM Test Register D-7
D.2.3 Clock Synthesizer Control Register D-7
D.2.4 Reset Status Register D-9
D.2.5 SCIM Test Register E D-9
D.2.6 Port A and B Data Registers D-9
D.2.7 Port G and H Data Registers D-10
D.2.8 Port G and H Data Direction Registers D-10
D.2.9 Port E Data Register D-10
D.2.10 Port E Data Direction Register D-11
D.2.11 Port E Pin Assignment Register D-11
D.2.12 Port F Data Register D-12
D.2.13 Port F Data Direction Register D-12
D.2.14 Port F Pin Assignment Register D-12
D.2.15 System Protection Control Register D-13
D.2.16 Periodic Interrupt Control Register D-15
D.2.17 Periodic Interrupt Timer Register D-15

TABLE OF CONTENTS
(Continued)

Paragraph **Title** **Page**

D.2.18 Software Watchdog Service Register D-16
D.2.19 Port F Edge-Detect Flag Register D-16
D.2.20 Port F Edge-Detect Interrupt Vector D-17
D.2.21 Port F Edge-Detect Interrupt Level D-17
D.2.22 Port C Data Register D-17
D.2.23 Chip-Select Pin Assignment Registers D-18
D.2.24 Chip-Select Base Address Register Boot D-20
D.2.25 Chip-Select Base Address Registers D-20
D.2.26 Chip-Select Option Register Boot D-21
D.2.27 Chip-Select Option Registers D-21
D.2.28 Master Shift Registers D-23
D.2.29 Test Module Shift Count Register D-24
D.2.30 Test Module Repetition Count Register D-24
D.2.31 Test Module Control Register D-24
D.2.32 Test Module Distributed Register D-24
D.3 Standby RAM Module D-25
D.3.1 RAM Module Configuration Register D-25
D.3.2 RAM Test Register D-26
D.3.3 Array Base Address Registers D-26
D.4 Masked ROM Module D-27
D.4.1 Masked ROM Module Configuration Register D-28
D.4.2 ROM Array Base Address Registers D-29
D.4.3 ROM Signature Registers D-31
D.4.4 ROM Bootstrap Words D-31
D.5 Analog-to-Digital Converter Module D-33
D.5.1 ADC Module Configuration Register D-34
D.5.2 ADC Test Register D-34
D.5.3 Port ADA Data Register D-34
D.5.4 Control Register 0 ..... D-35
D.5.5 Control Register 1 ..... D-36
D.5.6 Status Register D-40
D.5.7 Right Justified, Unsigned Result Register D-40
D.6 Multichannel Communication Interface Module D-42
D.6.1 MCCI Module Configuration Register D-42
D.6.2 MCCI Test Register D-43
D.6.3 SCI Interrupt Level Register D-43
D.6.4 MCCI Interrupt Vector Register D-44
D.6.5 SPI Interrupt Level Register D-44
D.6.6 MCCI Pin Assignment Register D-45
D.6.7 MCCI Data Direction Register D-46
D.6.8 MCCI Port Data Registers D-47

TABLE OF CONTENTS
(Continued)

Paragraph **Title** **Page**

D.6.9 SCI Control Register 0 ..... D-47
D.6.11 SCI Status Register D-50
D.6.12 SCI Data Register D-51
D.6.13 SPI Control Register D-52
D.6.14 SPI Status Register D-53
D.6.15 SPI Data Register D-54
D.7 Configurable Timer Module D-55
D.7.1 BIU Module Configuration Register D-57
D.7.2 BIUSM Test Configuration Register D-58
D.7.3 BIUSM Time Base Register D-58
D.7.4 CPSM Control Register D-58
D.7.5 CPSM Test Register D-59
D.7.6 MCSM Status/Interrupt/Control Registers D-59
D.7.7 MCSM Counter Registers D-61
D.7.8 MCSM Modulus Latch Registers D-62
D.7.9 FCSM Status/Interrupt/Control Register D-62
D.7.10 FCSM Counter Register D-64
D.7.11 DASM Status/Interrupt/Control Registers D-64
D.7.12 DASM Data Register A D-67
D.7.13 DASM Data Register B D-68
D.7.14 SASM Status/Interrupt/Control Registers D-69
D.7.15 SASM Data Registers D-72
D.7.16 PWM Status/Interrupt/Control Register D-73
D.7.17 PWM Period Registers D-76
D.7.18 PWM Pulse Width Registers D-77
D.7.19 PWM Counter Registers D-77
D.8 Flash EEPROM Modules D-79
D.8.1 Flash EEPROM Module Configuration Register D-80
D.8.2 Flash EEPROM Test Register D-82
D.8.3 Flash EEPROM Base Address Registers D-82
D.8.4 Flash EEPROM Control Register D-83
D.8.5 Flash EEPROM Bootstrap Words D-85
D.9 Block Erasable Flash D-86
D.9.1 BEFLASH Module Configuration Register D-86
D.9.2 BEFLASH Test Register D-87
D.9.3 BEFLASH Base Address Registers D-88
D.9.4 BEFLASH Control Register D-88
D.9.5 BEFLASH Bootstrap Words D-90

LIST OF ILLUSTRATIONS

Figure	Title	Page
3-1	MC68HC16R1 Block Diagram	3-4
3-2	MC68HC916R1 Block Diagram	3-5
3-3	MC68HC16R3 Block Diagram	3-6
3-4	MC68HC916R3 Block Diagram	3-7
3-5	MC68HC16R1 Pin Assignment for 132-Pin Package	3-8
3-6	MC68HC916R1 Pin Assignment for 132-Pin Package	3-9
3-7	MC68HC16R3 Pin Assignment for 132-Pin Package	3-10
3-8	MC68HC916R3 Pin Assignment for 132-Pin Package	3-11
3-9	Address Bus Connections Between the CPU16 and IMB	3-20
3-10	MC68HC16 R1 Address Map	3-22
3-11	MC68HC916R1 Address Map	3-23
3-12	MC68HC16 R3 Address Map	3-24
3-13	MC68HC916R3 Address Map	3-25
3-14	MC68HC16R1 Combined Program and Data Space Map	3-27
3-15	MC68HC16R1 Separate Program and Data Space Map	3-28
3-16	MC68HC916R1 Combined Program and Data Space Map	3-29
3-17	MC68HC916R1 Separate Program and Data Space Map	3-30
3-18	MC68HC16R3 Combined Program and Data Space Map	3-31
3-19	MC68HC16R3 Separate Program and Data Space Map	3-32
3-20	MC68HC916R3 Combined Program and Data Space Map	3-33
3-21	MC68HC916R3 Separate Program and Data Space Map	3-34
4-1	CPU16 Register Model	4-2
4-2	Condition Code Register	4-4
4-3	Data Types and Memory Organization	4-8
4-4	Basic Instruction Formats	4-34
4-5	Instruction Execution Model	4-35
4-7	BDM Serial I/O Block Diagram	4-45
4-8	BDM Connector Pinout	4-45
5-1	SCIM2 Block Diagram	5-2
5-2	Slow Reference Mode PLL Configuration	5-7
5-3	Slow Reference Mode Bypass/Filter Circuit	5-8
5-4	Slow Reference Crystal Circuit	5-9
5-5	Fast Reference Mode PLL Configuration	5-15
5-6	Fast Reference Mode Bypass/Filter Circuit	5-16
5-7	Fast Reference Crystal Circuit	5-16
5-8	External Clock Mode Divider Configuration	5-20
5-9	LPSTOP Flowchart	5-23
5-10	System Protection	5-24
5-11	Periodic Interrupt Timer and Software Watchdog Timer	5-27
5-12	MCU Basic System	5-31
5-13	Operand Byte Order	5-34

LIST OF ILLUSTRATIONS
(Continued)

Figure	Title	Page
5-14	Word Read Cycle Flowchart	5-38
5-15	Write Cycle Flowchart	5-39
5-16	CPU Space Address Encoding	5-41
5-17	Breakpoint Operation Flowchart	5-42
5-18	LPSTOP Interrupt Mask Encoding on DATA[15:0]	5-43
5-19	Bus Arbitration Flowchart for Single Request	5-47
5-20	Power-On Reset	5-52
5-21	Preferred Circuit for Data Bus Mode Select Conditioning	5-56
5-22	Alternate Circuit for Data Bus Mode Select Conditioning	5-57
5-23	Basic MCU System	5-67
5-24	Chip-Select Circuit Block Diagram	5-68
5-25	CPU Space Encoding for Interrupt Acknowledge	5-74
5-26	Port F Block Diagram	5-79
8-1	Programming Flowchart	8-5
8-2	Erasure Flowchart	8-7
8-3	Programming Voltage Envelope	8-8
8-4	V_{FPE} Conditioning Circuit	8-9
9-1	Programming Flowchart	9-5
9-2	Erasure Flowchart	9-8
9-3	Programming Voltage Envelope	9-9
9-4	V_{FPE} Conditioning Circuit	9-10
10-1	ADC Block Diagram	10-2
10-2	8-Bit Conversion Timing	10-12
10-3	10-Bit Conversion Timing	10-13
10-4	Analog Input Circuitry	10-15
10-5	Errors Resulting from Clipping	10-16
10-6	Star-Ground at the Point of Power Supply Origin	10-17
10-7	Input Pin Subjected to Negative Stress	10-19
10-8	Voltage LImiting Diodes in a Negative Stress Circuit	10-20
10-9	External Multiplexing Of Analog Signal Sources	10-21
10-10	Electrical Model of an A/D Input Pin	10-22
11-1	MCCI Block Diagram	11-1
11-2	SPI Block Diagram	11-5
11-3	CPHA = 0 SPI Transfer Format	11-9
11-4	CPHA = 1 SPI Transfer Format	11-10
11-5	SCI Transmitter Block Diagram	11-14
11-6	SCI Receiver Block Diagram	11-15
12-1	CTM7 Block Diagram	12-2
12-2	CTM8 Block Diagram	12-3
12-3	CPSM Block Diagram	12-7
12-4	FCSM Block Diagram	12-8

LIST OF ILLUSTRATIONS
(Continued)

Figure	Title	Page
12-5	MCSM Block Diagram	12-10
12-6	SASM Block Diagram	12-13
12-7	DASM Block Diagram	12-15
12-8	Pulse-Width Modulation Submodule Block Diagram	12-17
A-1	CLKOUT Output Timing Diagram	A-10
A-2	External Clock Input Timing Diagram	A-10
A-3	ECLK Output Timing Diagram	A-10
A-4	Read Cycle Timing Diagram	A-11
A-5	Write Cycle Timing Diagram	A-12
A-6	Fast Termination Read Cycle Timing Diagram	A-13
A-7	Fast Termination Write Cycle Timing Diagram	A-14
A-8	Bus Arbitration Timing Diagram — Active Bus Case	A-15
A-9	Bus Arbitration Timing Diagram — Idle Bus Case	A-16
A-10	Show Cycle Timing Diagram	A-17
A-11	Chip-Select Timing Diagram	A-18
A-12	Reset and Mode Select Timing Diagram	A-18
A-13	Background Debug Mode Timing Diagram (Serial Communication)	A-19
A-14	Background Debug Mode Timing Diagram (Freeze Assertion)	A-20
A-15	ECLK Timing Diagram	A-21
A-16	SPI Timing — Master, CPHA = 0	A-23
A-17	SPI Timing — Master, CPHA = 1	A-23
A-18	SPI Timing — Slave, CPHA = 0	A-24
A-19	SPI Timing — Slave, CPHA = 1	A-24
A-20	8-Bit ADC Conversion Accuracy	A-28
A-21	10-Bit ADC Conversion Accuracy	A-29
B-1	MC68HC16R1 Pin Assignment for 132-Pin Package	B-2
B-2	MC68HC916R1 Pin Assignment for 132-Pin Package	B-3
B-3	MC68HC16R3 Pin Assignment for 132-Pin Package	B-4
B-4	MC68HC916R3 Pin Assignment for 132-Pin Package	B-5
B-5	Case 831A-01 — 132-Pin Package Dimensions	B-6
D-1	CPU16 Register Model	D-2

LIST OF TABLES

Table	Title	Page
1-1	M68HC16 R-Series Modules	1-1
3-1	M68HC16 R-Series Pin Characteristics	3-12
3-2	M68HC16 R-Series Driver Types	3-15
3-3	M68HC16 R-Series Pin Functions	3-16
4-2	Instruction Set Summary	4-12
4-3	Instruction Set Abbreviations and Symbols	4-30
4-4	CPU16 Implementation of M68HC11 CPU Instructions	4-32
4-5	Exception Vector Table	4-38
4-6	IPIPE0/IPIPE1 Encoding	4-41
4-7	Command Summary	4-43
5-1	SCIMCR Noise Control Bits	5-3
5-2	Show Cycle Enable Bits	5-4
5-3	Effects of FREEZE Assertion	5-4
5-4	System Clock Sources	5-5
5-5	Slow Reference Mode Clock Control Multipliers	5-10
5-6	Slow Reference Mode Clock Frequencies	5-12
5-7	Fast Reference Mode Clock Control Multipliers	5-18
5-8	Fast Reference Mode Clock Frequencies	5-19
5-9	Bus Monitor Period	5-25
5-10	SWP Reset States	5-26
5-11	Software Watchdog Divide Ratio	5-27
5-12	PTP Reset States	5-28
5-13	Periodic Interrupt Priority	5-29
5-14	Size Signal Encoding	5-32
5-15	Address Space Encoding	5-32
5-16	Effect of DSACK Signals	5-34
5-17	Operand Alignment	5-36
5-18	DSACK[1:0], BERR, and HALT Assertion Results	5-44
5-19	Reset Source Summary	5-49
5-20	SCIM2 Pin States During Reset	5-53
5-21	Pins Associated with Basic Configuration Options	5-54
5-22	Operating Mode Selection	5-54
5-23	Address and Data Bus I/O Port Availability	5-54
5-24	Fully (16-bit) Expanded Mode Reset Configuration	5-59
5-25	Reset Pin Function of CS[10:6]	5-60
5-26	Reset Configuration for M68HC16 R-Series Memory Modules	5-60
5-27	Partially (8-bit) Expanded Mode Reset Configuration	5-61
5-28	Chip-Select Pin Functions	5-69
5-29	Pin Assignment Field Encoding	5-70
5-30	Block Size Encoding	5-71
5-31	Chip-Select Base and Option Register Reset Values	5-75

LIST OF TABLES
(Continued)

Table **Title** **Page**

5-32 CSBOOT Base and Option Register Reset Values 5-76
5-33 General-Purpose I/O Ports 5-76
5-34 Port E Pin Assignments 5-78
5-35 Port F Pin Assignments 5-80
5-36 PFPAR Pin Functions 5-80
6-1 SRAM Array Address Space Type 6-2
7-1 ROM Array Space Field 7-2
7-2 Wait States Field 7-3
8-1 Wait States Field 8-3
9-1 Block-Erasable Flash Sectoring 9-1
9-2 BEFLASH Erase Operation Address Ranges 9-6
10-1 FRZ Field Encoding 10-4
10-2 Multiplexer Channel Sources 10-5
10-3 Sample Time Selection 10-7
10-4 Conversion Parameters Controlled by ADCTL1 10-8
10-5 ADC Conversion Modes 10-8
10-6 Single-Channel Conversions (MULT = 0) 10-10
10-7 Multiple-Channel Conversions (MULT = 1) 10-11
10-8 Result Register Formats 10-14
10-9 External Circuit Settling Time (10-Bit Conversions) 10-24
10-10 Error Resulting From Input Leakage (IOFF) 10-24
11-1 MCCI Interrupt Vectors 11-3
11-2 Pin Assignments 11-4
11-3 SPI Pin Functions 11-7
11-4 SCK Frequencies 11-11
11-5 SCI Pins 11-17
11-6 Serial Frame Formats 11-18
11-7 Effect of Parity Checking on Data Size 11-19
12-1 CTM Time Base Bus Allocation 12-5
12-2 SASM Operational Modes 12-12
12-3 DASM Modes of Operation 12-14
12-4 Channel B Data Register Access 12-15
12-5 PWMSM Divide By Options 12-18
12-6 PWM Pulse and Frequency Ranges (in Hz) Using ÷ 2 Option (16.78 MHz) ... 12-21
12-7 PWM Pulse and Frequency Ranges (in Hz) Using ÷ 3 Option (16.78 MHz) ... 12-21
12-8 CTM7 Interrupt Priority and Vector/Pin Allocation 12-23
12-9 CTM8 Interrupt Priority and Vector/Pin Allocation 12-24
A-1 Maximum Ratings A-1
A-2 Typical Ratings A-2
A-3 Thermal Characteristics A-3
A-4 Clock Control Timing A-4

LIST OF TABLES
(Continued)

Table **Title** **Page**

A-5 DC Characteristics A-5
A-6 AC Timing A-7
A-7 20.97 MHz Background Debug Mode Timing A-19
A-8 ECLK Bus Timing A-20
A-9 SPI Timing A-22
A-10 ADC Maximum Ratings A-25
A-11 ADC DC Electrical Characteristics (Operating) A-26
A-12 ADC AC Characteristics (Operating) A-26
A-13 ADC Conversion Characteristics (Operating) A-27
A-14 BEFLASH/Flash EEPROM Module Specifications A-30
A-15 BEFLASH/Flash EEPROM Module Life A-30
D-1 Module Address Map D-1
D-2 SCIM2 Address Map D-4
D-3 Show Cycle Enable Bits D-6
D-4 Port E Pin Assignments D-11
D-5 Port F Pin Assignments D-13
D-6 PFPAR Pin Functions D-13
D-7 Software Watchdog Divide Ratio D-14
D-8 Bus Monitor Period D-15
D-9 Pin Assignment Field Encoding D-18
D-10 CSPAR0 Pin Assignments D-18
D-11 CSPAR1 Pin Assignments D-19
D-12 Reset Pin Function of CS[10:6] D-19
D-13 Block Size Encoding D-20
D-14 BYTE Field Bit Encoding D-21
D-15 Read/Write Field Bit Encoding D-21
D-16 DSACK Field Encoding D-22
D-17 Address Space Bit Encodings D-23
D-18 Interrupt Priority Level Field Encoding D-23
D-19 SRAM Address Map D-25
D-20 SRAM Array Address Space Type D-26
D-21 MRM Address Map D-27
D-22 MRxMCR Registers D-28
D-23 ROM Array Space Field D-29
D-24 Wait States Field D-29
D-25 ROMxBAH and ROMxBAL Registers D-30
D-26 RxSIGHI and RxSIGLO Registers D-31
D-27 ROMxBS[0:3] Registers D-32
D-28 ADC Module Address Map D-33
D-29 Freeze Encoding D-34
D-30 Sample Time Selection D-35

LIST OF TABLES
(Continued)

Table **Title** **Page**

D-31 Prescaler Output D-36
D-32 ADC Conversion Mode D-37
D-33 Single-Channel Conversions (MULT = 0) D-38
D-34 Multiple-Channel Conversions (MULT = 1) D-39
D-35 MCCI Address Map D-42
D-36 Interrupt Vector Sources D-44
D-37 MPAR Pin Assignments D-45
D-38 Effect of MDDR on MCCI Pin Function D-46
D-39 Examples of SCI Baud Rates D-48
D-40 CTM Address Map D-55
D-41 Interrupt Vector Base Number Bit Field D-57
D-42 Time Base Register Bus Select Bits D-58
D-43 Prescaler Division Ratio Select Field D-59
D-44 MCSMxSIC Registers D-59
D-45 Drive Time Base Bus Field D-60
D-46 Modulus Load Edge Sensitivity Bits D-61
D-47 Counter Clock Select Field D-61
D-48 MCSMxCNT Registers D-61
D-49 MCSMxML Registers D-62
D-50 Drive Time Base Bus Field D-63
D-51 Counter Clock Select Field D-63
D-52 DASMxSIC Registers D-64
D-53 DASM Mode Flag Status Bit States D-65
D-54 Edge Polarity D-66
D-55 DASM Mode Select Field D-67
D-56 DASMxA Registers D-67
D-57 DASMA Operations D-68
D-58 DASMxB Registers D-68
D-59 DASMB Operations D-69
D-60 SICxA Registers D-69
D-61 SICxB Registers D-70
D-62 Event Flag Status Conditions D-70
D-63 SASM Operating Mode Select D-72
D-64 SxDATA Registers D-73
D-65 SxDATB Registers D-73
D-66 PWMxSIC Registers D-74
D-67 PWMSM Output Pin Polarity Selection D-75
D-68 PWMSM Divide By Options D-76
D-69 PWxMA1 Registers D-76
D-70 PWMxB1 Registers D-77
D-71 PWMxC Registers D-78

LIST OF TABLES
(Continued)

Table **Title** **Page**

D-72 Flash EEPROM Address Map....D-79
D-73 FEExMCR Registers....D-80
D-74 Array Space Encoding....D-81
D-75 Wait State Encoding....D-82
D-76 FEExTST Registers....D-82
D-77 FEExBAH Registers....D-83
D-78 FEExBAL Registers....D-83
D-79 FEExCTL Registers....D-84
D-80 FEExBS[3:0] Words....D-85
D-81 Bootstrap Words....D-85
D-82 BEFLASH Address Map....D-86
D-83 Array Space Encoding....D-87
D-84 BEFLASH Erase Operation Address Ranges....D-89

SECTION 1
INTRODUCTION

M68HC16 R-series microcontrollers (including the MC68HC16R1, MC68HC916R1, MC68HC16R3, and the MC68HC916R1) are high-speed 16-bit control units that are upwardly code compatible with M68HC11 controllers. Both are members of the M68HC16 family of modular microcontrollers.

M68HC16 microcontroller units (MCUs) are built up from standard modules that interface via a common internal bus. Standardization facilitates rapid development of devices tailored for specific applications.

M68HC16 R-series MCUs incorporate a number of different modules. Refer to **Table 1-1** for the modules that are present on a particular MCU. (X) indicates that the module is present on the MCU. All of these modules are interconnected by the intermodule bus (IMB).

Table 1-1 M68HC16 R-Series Modules

Modules	MC68HC16R1	MC68HC916R1	MC68HC16R3	MC68HC916R3
Central Processor Unit (CPU16)	X	X	X	X
Single-Chip Integration Module 2 (SCIM2)	X	X	X	X
Standby RAM (SRAM)	2 Kbytes	2 Kbytes	4 Kbytes	4 Kbytes
Masked ROM Module(s) (MRM)	48 Kbytes	—	32 Kbytes 46 Kbytes	—
Flash EEPROM Modules	—	16 Kbytes + 32 Kbytes	—	3 X 32 Kbytes
Block Erasable Flash EEPROM	—	2 Kbytes	—	2 Kbytes
Analog-to-Digital Converter (ADC)	X	X	X	X
Multichannel Communication Interface (MCCI)	X	X	X	X
Configurable Timer Module (CTM)	CTM7	CTM7	CTM8	CTM8

The maximum operating frequency for M68HC16 R-series MCUs is 16.78 MHz. An internal phase-locked loop circuit can synthesize the system clock from either a slow (typically 32.768 kHz) or fast (typically 4.194 MHz) reference, or an externally generated frequency can be used for the system clock. System hardware and software support changes in clock rate during operation. Because the MCU designs are fully static, register and memory contents are not affected by clock rate changes.

High-density complementary metal-oxide semiconductor (HCMOS) architecture makes the basic power consumption low. Power consumption can be minimized by stopping the system clock. The M68HC16 instruction set includes a low-power stop (LPSTOP) command that efficiently implements this capability.

Documentation for the Modular Microcontroller Family follows the modular construction of the devices in the product line. Each device has a comprehensive user's manual that provides sufficient information for normal operation of the device. The user's manual is supplemented by module reference manuals that provide detailed information about module operation and applications. Refer to individual module chapters for information on additional documentation.

SECTION 2
NOMENCLATURE

The following tables show the nomenclature used throughout the M68HC16 R-Series User's Manual.

2.1 Symbols and Operators

Symbol	Function
+	Addition
-	Subtraction (two's complement) or negation
*	Multiplication
/	Division
>	Greater
<	Less
=	Equal
≥	Equal or greater
≤	Equal or less
≠	Not equal
•	AND
;	Inclusive OR (OR)
⊕	Exclusive OR (EOR)
NOT	Complementation
:	Concatenation
⇒	Transferred
⇔	Exchanged
±	Sign bit; also used to show tolerance
«	Sign extension
%	Binary value
$	Hexadecimal value

2.2 CPU16 Register Mnemonics

Mnemonic	Register
A	Accumulator A
AM	Accumulator M
B	Accumulator B
CCR	Condition code register
D	Accumulator D
E	Accumulator E
EK	Extended addressing extension field
HR	MAC multiplier register
IR	MAC multiplicand register
IX	Index register X
IY	Index register Y
IZ	Index register Z
K	Address extension register
PC	Program counter
PK	Program counter extension field
SK	Stack pointer extension field
SP	Stack pointer
XK	Index register X extension field
YK	Index register Y extension field
ZK	Index register Z extension field
XMSK	Modulo addressing index register X mask
YMSK	Modulo addressing index register Y mask
S	LPSTOP mode control bit
MV	AM overflow flag
H	Half carry flag
EV	AM extended overflow flag
N	Negative flag
Z	Zero flag
V	Two’s complement overflow flag
C	Carry/borrow flag
IP	Interrupt priority field
SM	Saturation mode control bit

2.3 Register Mnemonics

Mnemonic	Register
ADCMCR	ADC Module Configuration Register
ADTEST	ADC Test Register
ADCTL[0:1]	ADC Control Registers [0:1]
ADSTAT	ADC Status Register
BFEBAH	BEFLASH Base Address High Register
BFEBAL	BEFLASH Base Address Low Register
BFEBS[0:3]	BEFLASH Bootstrap Words [0:3]
BFECTL	BEFLASH Control Register
BFEMCR	BEFLASH Module Configuration Register
BFETST	BEFLASH Test Register
BIUMCR	CTM7/8 BIUSM Module Configuration Register (R Series)
BIUTEST	CTM7/8 BIUSM Test Register (R Series)
BIUTBR	CTM7/8 BIUSM Time Base Register (R Series)
CPCR	CTM7/8 CPSM Control Register (R Series)
CPTR	CTM7/8 CPSM Test Register (R Series)
CREG	SCIM2 Test Module Control Register
CSBARBT	SCIM2 Chip-Select Base Address Register Boot
CSBAR[0:10]	SCIM2 Chip-Select Base Address Registers [0:10]
CSORBT	SCIM2 Chip-Select Option Register Boot
CSOR[0:10]	SCIM2 Chip-Select Option Registers [0:10]
CSPAR[0:1]	SCIM2 Chip-Select Pin Assignment Registers [0:1]
DASM[4:5]A	CTM7 DASM A Data Registers [4:5] (R1/916R1)
DASM[4:5]B	CTM7 DASM B Data Registers [4:5] (R1/916R1)
DASM[4:5]SIC	CTM7 DASM Status/Interrupt/Control Registers [4:5] (R1/916R1)
DASM[20]/[22]A	CTM8 DASM A Data Register [20]/[22] (R3/916R3)
DASM[20]/[22]B	CTM8 DASM B Data Register [20]/[22] (R3/916R3)
DASM[20]/[22]SIC	CTM8 DASM Status/Interrupt/Control Register [20]/[22] (R3/916R3)
DDRAB	SCIM2 Port A/B Data Direction Register
DDRE	SCIM2 Port E Data Direction Register
DDRF	SCIM2 Port F Data Direction Register
DDRG	SCIM2 Port G Data Direction Register
DDRH	SCIM2 Port H Data Direction Register
DDRM	MCCI Data Direction Register
DREG	SCIM2 Test Module Distributed Register
FCSM3SIC	CTM7/8 FCSM3 Status/Interrupt/Control Register (R Series)
FCSM3CNT	CTM7/8 FCSM3 Counter Register (R Series)
FEE[1:2]BAH	Flash EEPROM Base Address High Registers [1:2] (916R1)
FEE[1:3]BAH	Flash EEPROM Base Address High Registers [1:3] (916R3)
FEE[1:2]BAL	Flash EEPROM Base Address Low Registers [1:2] (916R1)
FEE[1:3]BAL	Flash EEPROM Base Address Low Registers [1:3] (916R3)
FEE[1:2]BS[0:3]	Flash EEPROM [1:2] Bootstrap Words [0:3] (916R1)

Mnemonic	Register
FEE[1:3]BS[0:3]	Flash EEPROM [1:3] Bootstrap Words [0:3] (916R3)
FEE[1:2]CTL	Flash EEPROM Control Registers [1:2] (916R1)
FEE[1:3]CTL	Flash EEPROM Control Registers [1:3] (916R3)
FEE[1:2]MCR	Flash EEPROM Module Configuration Registers [1:2] (916R1)
FEE[1:3]MCR	Flash EEPROM Module Configuration Registers [1:3] (916R3)
FEE[1:2]TST	Flash EEPROM Test Registers [1:2] (916R1)
FEE[1:3]TST	Flash EEPROM Test Registers [1:3] (916R3)
ILSCI	MCCI SCI Interrupt Level Register
ILSPI	MCCI SPI Interrupt Level Register
LJSRR[0:7]	ADC Left-Justified Signed Result Registers [0:7]
LJURR[0:7]	ADC Left-Justified Unsigned Result Registers [0:7]
MCSM2SIC	CTM7 MCSM2 Status/Interrupt/Control Register (R1/916R1)
MCSM2CNT	CTM7 MCSM2 Counter Register (R1/916R1)
MCSM2ML	CTM7 MCSM2 Modulus Latch (R1/916R1)
MCSM[21]/[23]SIC	CTM8 MCSM[21]/[23] Status/Interrupt/Control Register (R3/916R3)
MCSM[21]/[23]CNT	CTM8 MCSM[21]/[23] Counter Register (R3/916R3)
MCSM[21]/[23]ML	CTM8 MCSM[21]/[23] Modulus Latch (R3/916R3)
MIVR	MCCI Interrupt Vector Register
MMCR	MCCI Module Configuration Register
MPAR	MCCI Pin Assignment Register
MRMCR	Masked ROM Module Configuration Register
MTEST	MCCI Test Register
PEPAR	SCIM2 Port E Pin Assignment Register
PFIVR	SCIM2 Port F Edge Detect Interrupt Vector
PFLVR	SCIM2 Port F Edge Detect Interrupt Level
PFPAR	SCIM2 Port F Pin Assignment Register
PICR	SCIM2 Periodic Interrupt Control Register
PITR	SCIM2 Periodic Interrupt Timer Register
PORTA	SCIM2 Port A Data Register
PORTADA	ADC Port ADA Data Register
PORTB	SCIM2 Port B Data Register
PORTC	SCIM2 Port C Data Register
PORTE[0:1]	SCIM2 Port E Data Registers [0:1]
PORTF[0:1]	SCIM2 Port F Data Registers [0:1]
PORTG	SCIM2 Port G Data Register
PORTH	SCIM2 Port H Data Register
PORTFE	SCIM2 Port F Edge Detect Flag
PORTMC	MCCI Port Data Register
PORTMCP	MCCI Port Pin State Register
PWM[18:19]A	CTM7/8 PWSM Period [18:19] (R Series)
PWM[18:19]B	CTM7/8 PWSM Pulse Width [18:19] (R Series)
PWM[18:19]C	CTM7/8 PWSM Counter [18:19] (R Series)
PWM[18:19]SIC	CTM7/8 PWSM Status/Interrupt/Control Registers [18:19] (R Series)

Mnemonic	Register
RAMBAH	RAM Array Base Address High Register
RAMBAL	RAM Array Base Address Low Register
RAMMCR	RAM Module Configuration Register
RAMTST	RAM Test Register
RJURR[0:7]	ADC Right-Justified Unsigned Result Registers [0:7]
ROMBAH	ROM Base Address High Register
ROMBAL	ROM Base Address Low Register
ROMBS[0:3]	ROM Bootstrap Words [0:3]
RSR	SCIM2 Reset Status Register
SCCR[0:1]	SCI Control Registers [0:1]
SCDR	SCI Data Register
SCSR	SCI Status Register
SCIM2CR	SCIM2 Module Configuration Register
SCIM2TR	SCIM2 Test Register
SCIM2TRE	SCIM2 Test Register E
SIC[6]/[8]/[10]/[12]/[14]/[16]A	CTM7/8 SASM A Status/Interrupt/Control Registers [6]/[8]/[10]/[12]/[14]/[16] (R Series)
SIC[6]/[8]/[10]/[12]/[14]/[16]B	CTM7/8 SASM B Status/Interrupt/Control Registers [6]/[8]/[10]/[12]/[14]/[16] (R Series)
S[6]/[8]/[10]/[12]/[14]/[16]DATA	CTM7/8 SASM A Data Registers [6]/[8]/[10]/[12]/[14]/[16] (R Series)
S[6]/[8]/[10]/[12]/[14]/[16]DATB	CTM7/8 SASM B Data Registers [6]/[8]/[10]/[12]/[14]/[16] (R Series)
SIGHI	ROM Signature High Register
SIGLO	ROM Signature Low Register
SCCR0[A:B]	MCCI SCI Control 0 Registers [A:B]
SCCR1[A:B]	MCCI SCI Control 1 Registers [A:B]
SCDR[A:B]	MCCI SCI Data Registers [A:B]
SCDR[A:B]	MCCI SCI Status Registers [A:B]
SPCR	MCCI SPI Control Register
SPDR	MCCI SPI Data Register
SPSR	MCCI SPI Status Register
SWSR	SCIM2 Software Watchdog Service Register
SYNCR	SCIM2 Clock Synthesizer Control Register
SYPCR	SCIM2 System Protection Control Register
TCNT	SCIM2 Timer Counter Register
TSTMSRA	SCIM2 Test Master Shift Register A
TSTMSRB	SCIM2 Test Master Shift Register B
TSTRC	SCIM2 Test Repetition Count Register
TSTSC	SCIM2 Test Shift Count Register

2.4 Conventions

Logic level one is the voltage that corresponds to a Boolean true (1) state.

Logic level zero is the voltage that corresponds to a Boolean false (0) state.

Set refers specifically to establishing logic level one on a bit or bits.

Clear refers specifically to establishing logic level zero on a bit or bits.

Asserted means that a signal is in active logic state. An active low signal changes from logic level one to logic level zero when asserted, and an active high signal changes from logic level zero to logic level one.

Negated means that an asserted signal changes logic state. An active low signal changes from logic level zero to logic level one when negated, and an active high signal changes from logic level one to logic level zero.

A specific mnemonic within a range is referred to by mnemonic and number. A15 is bit 15 of Accumulator A; ADDR7 is line 7 of the address bus; CSOR0 is chip-select option register 0. **A range of mnemonics** is referred to by mnemonic and the numbers that define the range. VBR[4:0] are bits four to zero of the Vector Base Register; CSOR[0:5] are the first six chip-select option registers.

Parentheses are used to indicate the content of a register or memory location, rather than the register or memory location itself. For example, (A) is the content of Accumulator A. (M : M + 1) is the content of the word at address M.

LSB means least significant bit or bits. **MSB** means most significant bit or bits. References to low and high bytes are spelled out.

LSW means least significant word or words. **MSW** means most significant word or words.

ADDR is the address bus. ADDR[7:0] are the eight LSB of the address bus.

DATA is the data bus. DATA[15:8] are the eight MSB of the data bus.

SECTION 3
OVERVIEW

This section provides general information on the M68HC16 R-series MCUs. It lists the features of each of the modules, shows device functional divisions and pinouts, summarizes signal and pin functions, discusses the intermodule bus, and provides system memory maps. Timing and electrical specifications for the entire microcontroller and for individual modules are provided in APPENDIX A ELECTRICAL CHARACTERISTICS. Comprehensive module register descriptions and memory maps are provided in **APPENDIX D REGISTER SUMMARY**.

3.1 M68HC16 R-Series MCU Features

The following paragraphs highlight capabilities of each of the MCU modules. Each module is discussed separately in a subsequent section of this manual.

3.1.1 Central Processor Unit (CPU16)

- 16-Bit architecture
- Full set of 16-bit instructions
- Three 16-bit index registers
- Two 16-bit accumulators
- Control-oriented digital signal processing capability
- Addresses up to 1 Mbyte of program memory; 1 Mbyte of data memory
- Background debug mode
- Fully static operation

3.1.2 Single-Chip Integration Module 2 (SCIM2)

- Single-chip and expanded operating modes
- External bus support in expanded mode
- Nine programmable chip-select outputs
- Phase-locked loop system clock with user-selectable fast or slow reference
- Watchdog timer, clock monitor, and bus monitor
- Address and data bus provide 32 discrete I/O lines in single-chip mode
- Enhanced reset controller

3.1.3 Standby RAM (SRAM)

- The MC68HC16R1 and the MC68HC916R1 have 2 Kbytes of static RAM
- The MC68HC16R3 and the MC68HC916R3 have 4 Kbytes of static RAM
- Standby voltage (V_{STBY}) input for low-power standby operation of the SRAM
- Power-down status flag denotes loss of V_{STBY} during low-power standby operation

3.1.4 Masked ROM Module (MRM)

- The MC68HC16R1 has a 48-Kbyte array, accessible as bytes or words
- The MC68HC16R3 has 32-Kbyte and 64-Kbyte arrays, accessible as bytes or words
- User selectable default base address
- User selectable bootstrap ROM function
- User selectable ROM verification code

3.1.5 Flash EEPROM Modules (FLASH)

- The MC68HC916R1 has 16-Kbyte and 32-Kbyte arrays
- The MC68HC916R3 has three 32-Kbyte arrays
- Bulk erase and byte/word program with 12 volt external program/erase voltage
- Modules can be mapped to provide 48 Kbytes (for the MC68HC916R1) or 96-Kbytes (for the MC68HC916R3) of contiguous address space

3.1.6 Block Erasable Flash EEPROM (BEFLASH)

- The MC68HC916R1 and the MC68HC916R3 both have a 2-Kbyte block erasable flash EEPROM
- Bulk/block erase and byte/word program with 12 volt external program/erase voltage

3.1.7 Analog-to-Digital Converter (ADC)

- Eight channels, eight result registers, three result alignment modes
- Eight automated modes

3.1.8 Multichannel Communication Interface (MCCI)

- Two channels of enhanced SCI (UART)
- One channel of SPI

3.1.9 Configurable Timer Module (CTM)

- The MC68HC16R1 and the MC68HC916R1 have the CTM7 which includes :
  - — One 16-bit free-running counter submodule (FCSM)
  - — One 16-bit modulus counter submodule (MCSM)
  - — Six single-action submodules (SASMs)
  - — Two double-action submodules (DASMs)
  - — Two pulse-width submodules (PWMSMs)
  - — One external clock pin for the modulus and free-running counter submodules
- The MC68HC16R3 and the MC68HC916R3 have the CTM8 which includes:
  - — One 16-bit free-running counter submodule (FCSM)
  - — Three 16-bit modulus counter submodules (MCSMs)
  - — Six single-action submodules (SASMs)
  - — Two double-action submodules (DASMs)
  - — Two pulse-width submodules (PWMSMs)
  - — One external clock pin for the modulus submodules

3.2 Intermodule Bus

The intermodule bus (IMB) is a standardized bus developed to facilitate the design and operation of modular microcontrollers. It contains circuitry that supports exception processing, address space partitioning, multiple interrupt levels, and vectored interrupts. The standardized modules in the MCU communicate with one another through the IMB. Although the full IMB supports 24 address and 16 data lines, CPU16-based MCUs use only 20 address lines. ADDR[23:20] follow the state of ADDR19.

3.3 System Block Diagram and Pin Assignment Diagrams

Figures 3-1 through **3-4** show functional block diagrams of M68HC16 R-series MCUs. Although diagram blocks represent the relative location of the physical modules, there is not a one-to-one correspondence between the location and size of blocks in the diagram and the location and size of modules on the integrated circuit.

Figures 3-5 through **3-8** shows the pin assignments for M68HC16 R-series MCUs in a 132-pin plastic surface-mount package. Refer to APPENDIX B MECHANICAL DATA AND ORDERING INFORMATION for package dimensions. Refer to subsequent paragraphs in this section for pin and signal descriptions.

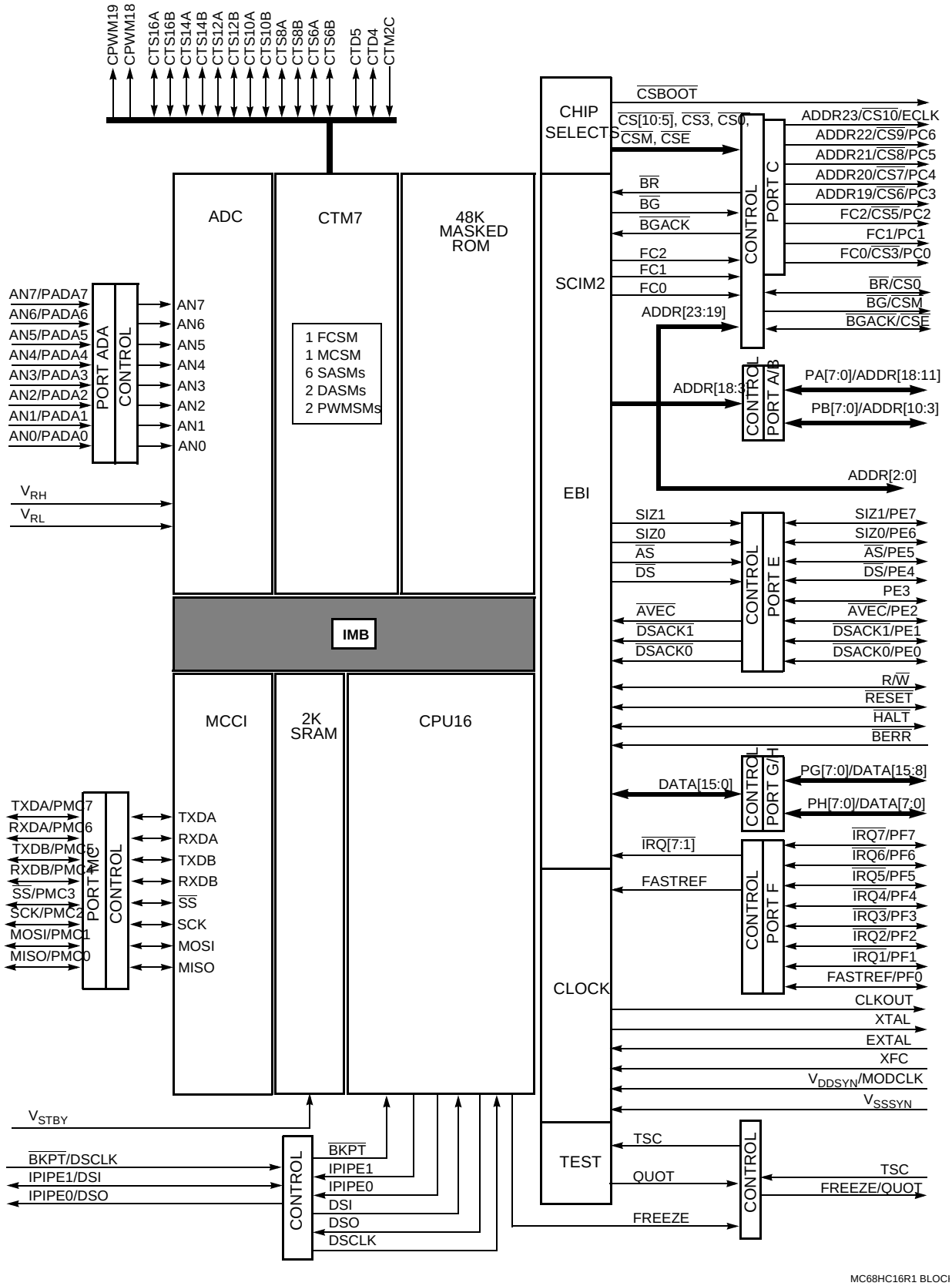


Figure 3-1 MC68HC16R1 Block Diagram

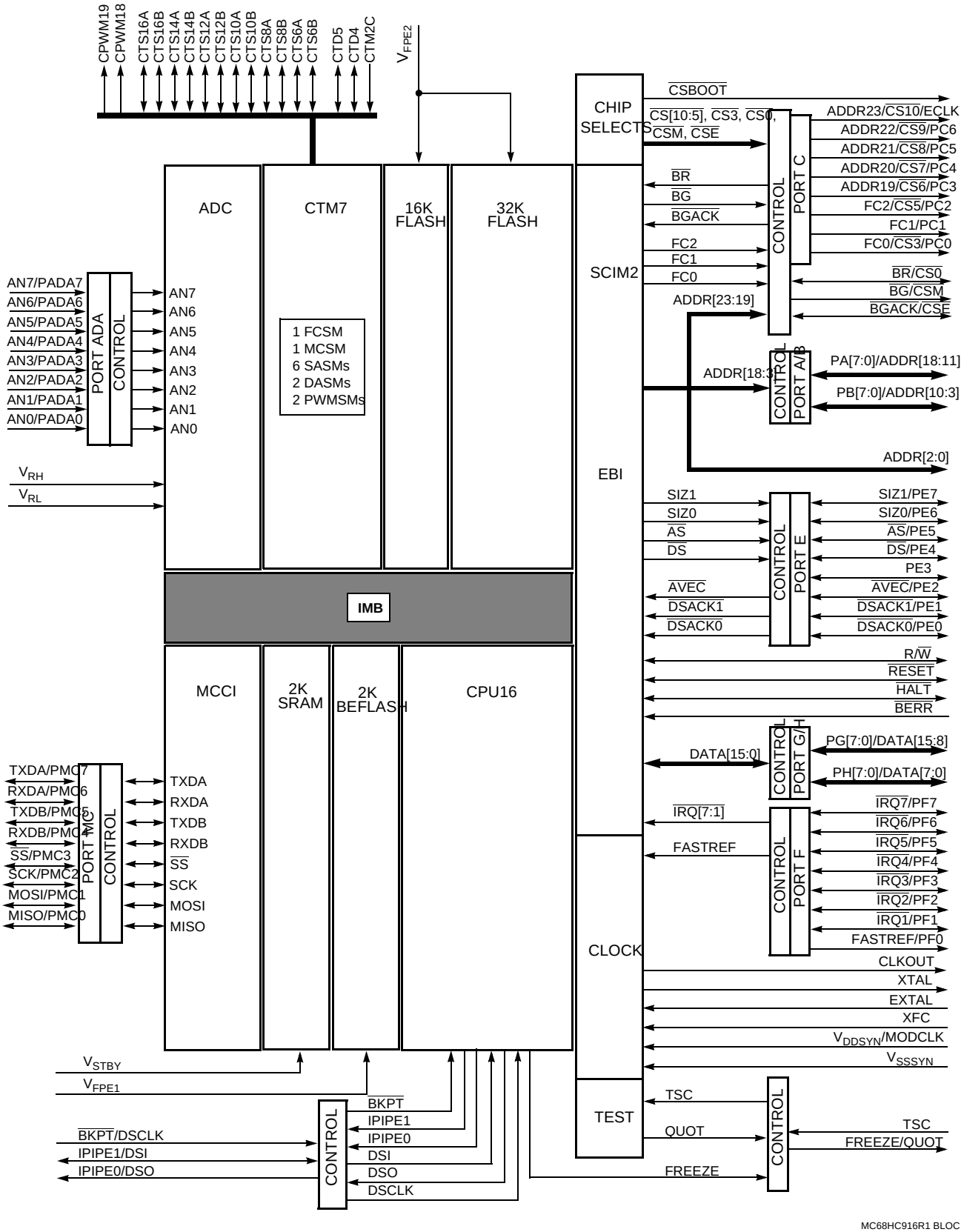


Figure 3-2 MC68HC916R1 Block Diagram

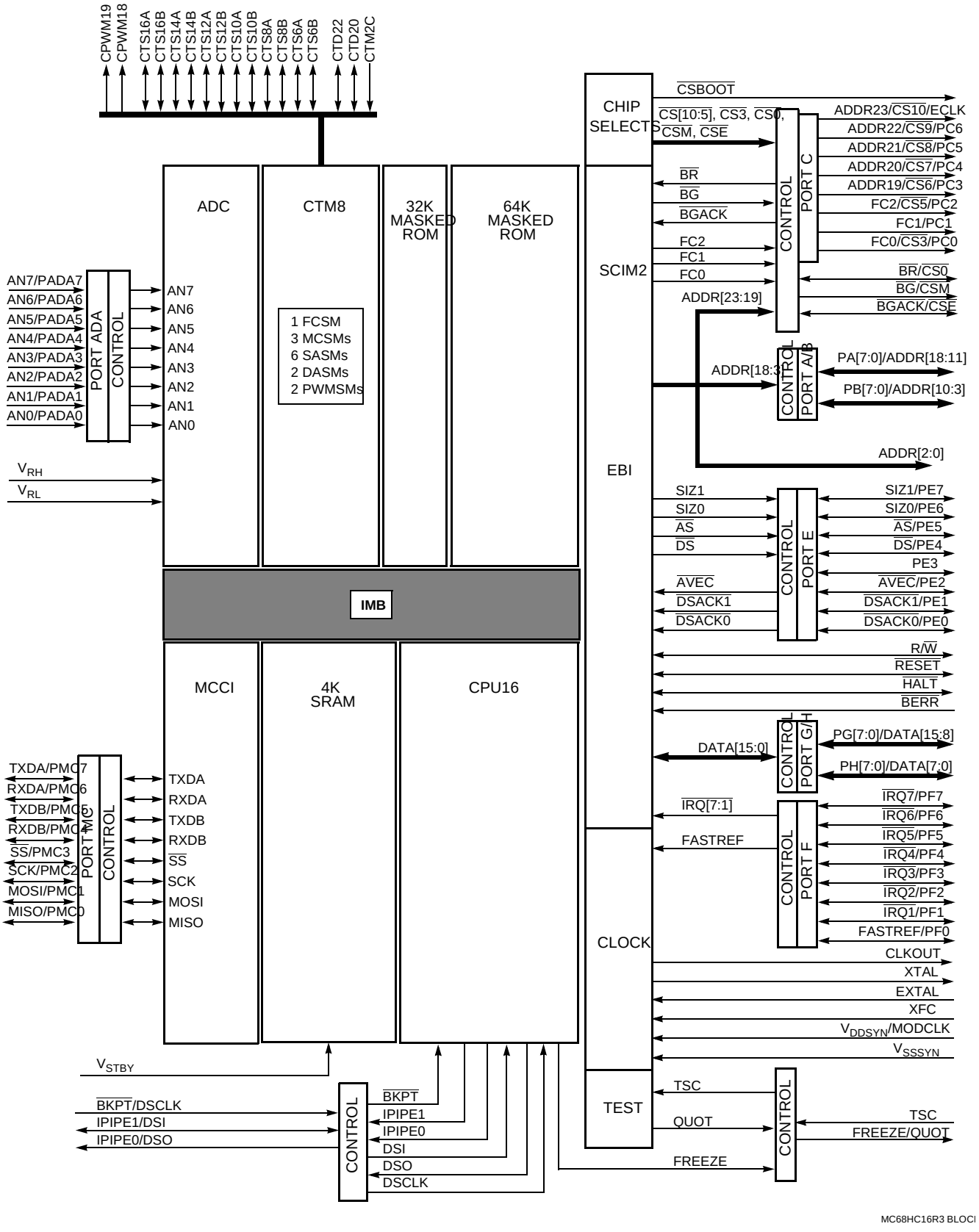


Figure 3-3 MC68HC16R3 Block Diagram

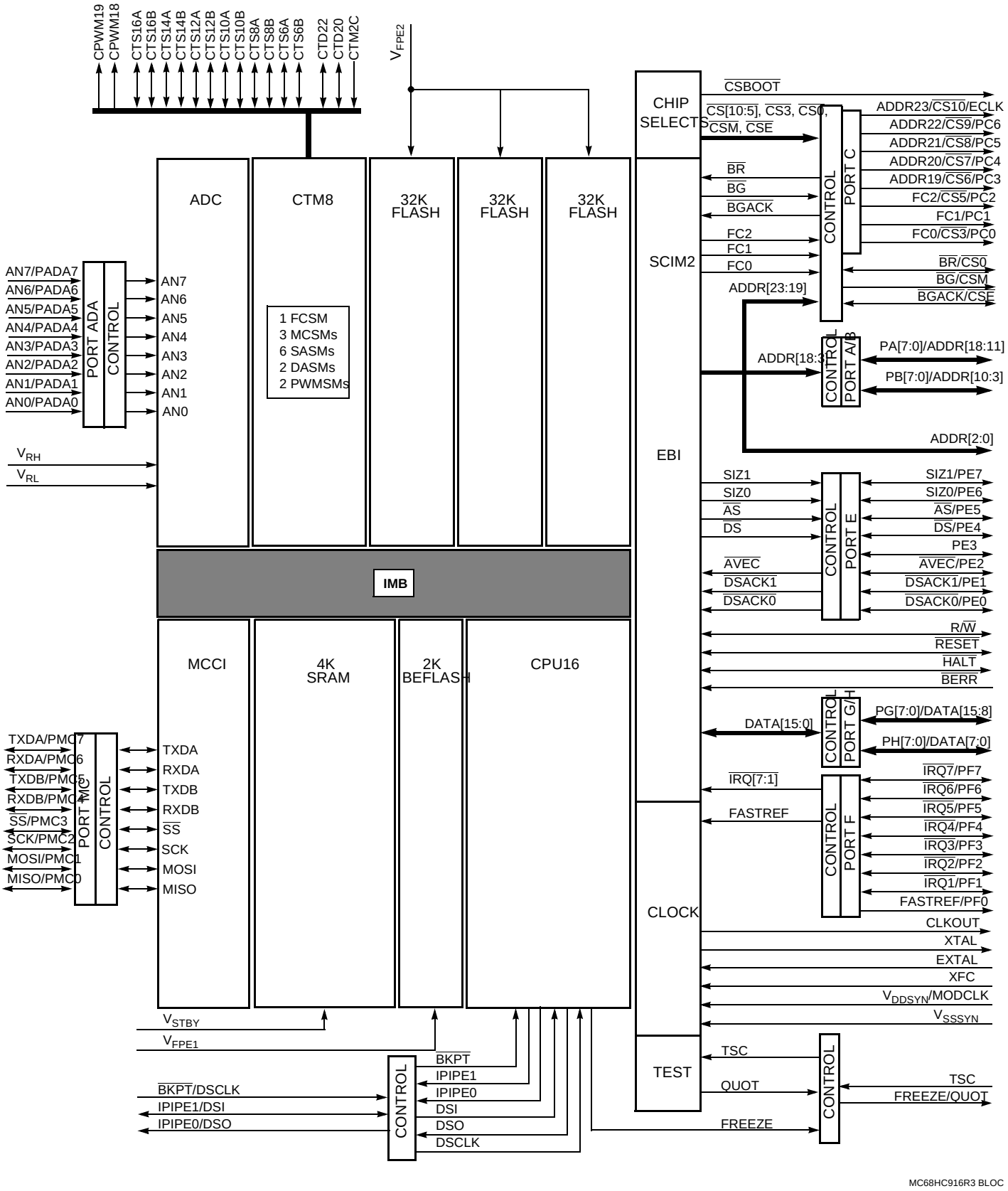
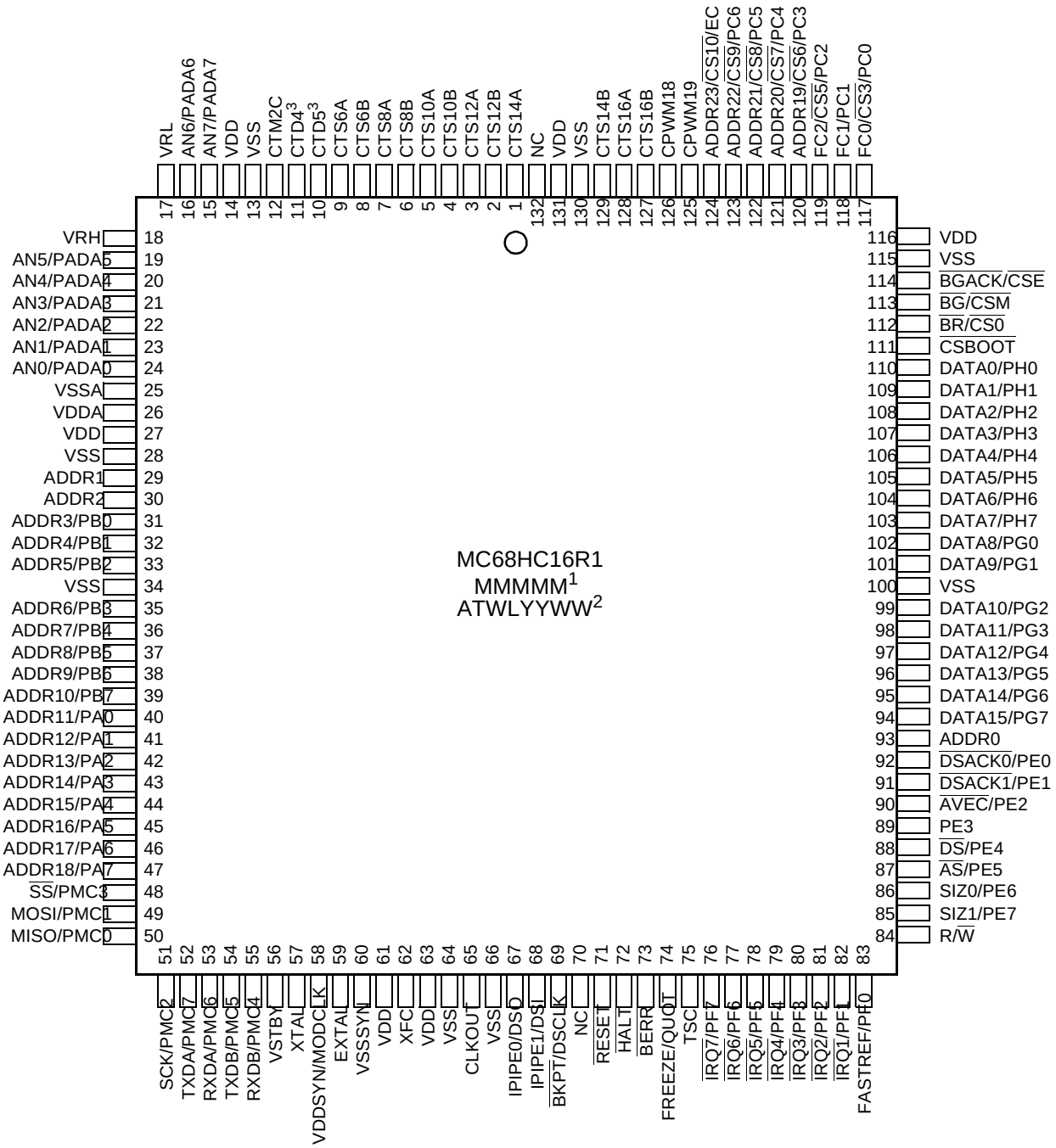


Figure 3-4 MC68HC916R3 Block Diagram

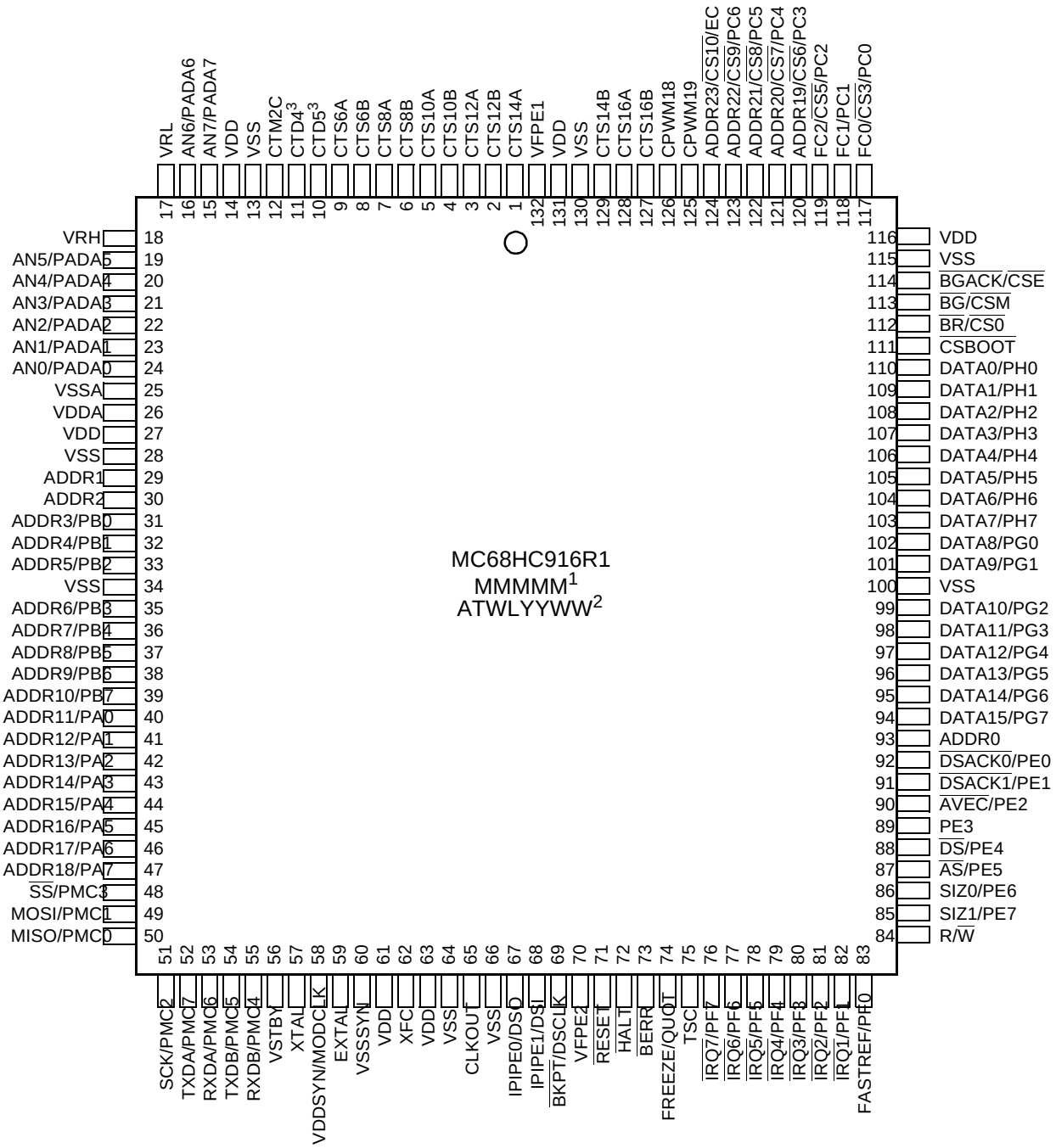


NOTES:

1. MMMMM = MASK OPTION NUMBER
2. ATWLYYWW = ASSEMBLY TEST LOCATION/YEAR, WEEK
3. R-SERIES MCUS ARE COMPATIBLE. THE ONLY DIFFERENCE IS IN THE NAMING OF THE CTM7 AND CTM8 DOUBLE-ACTION SUBMODULE PINS.

MC68HC16R1 132-PIN QF

Figure 3-5 MC68HC16R1 Pin Assignment for 132-Pin Package

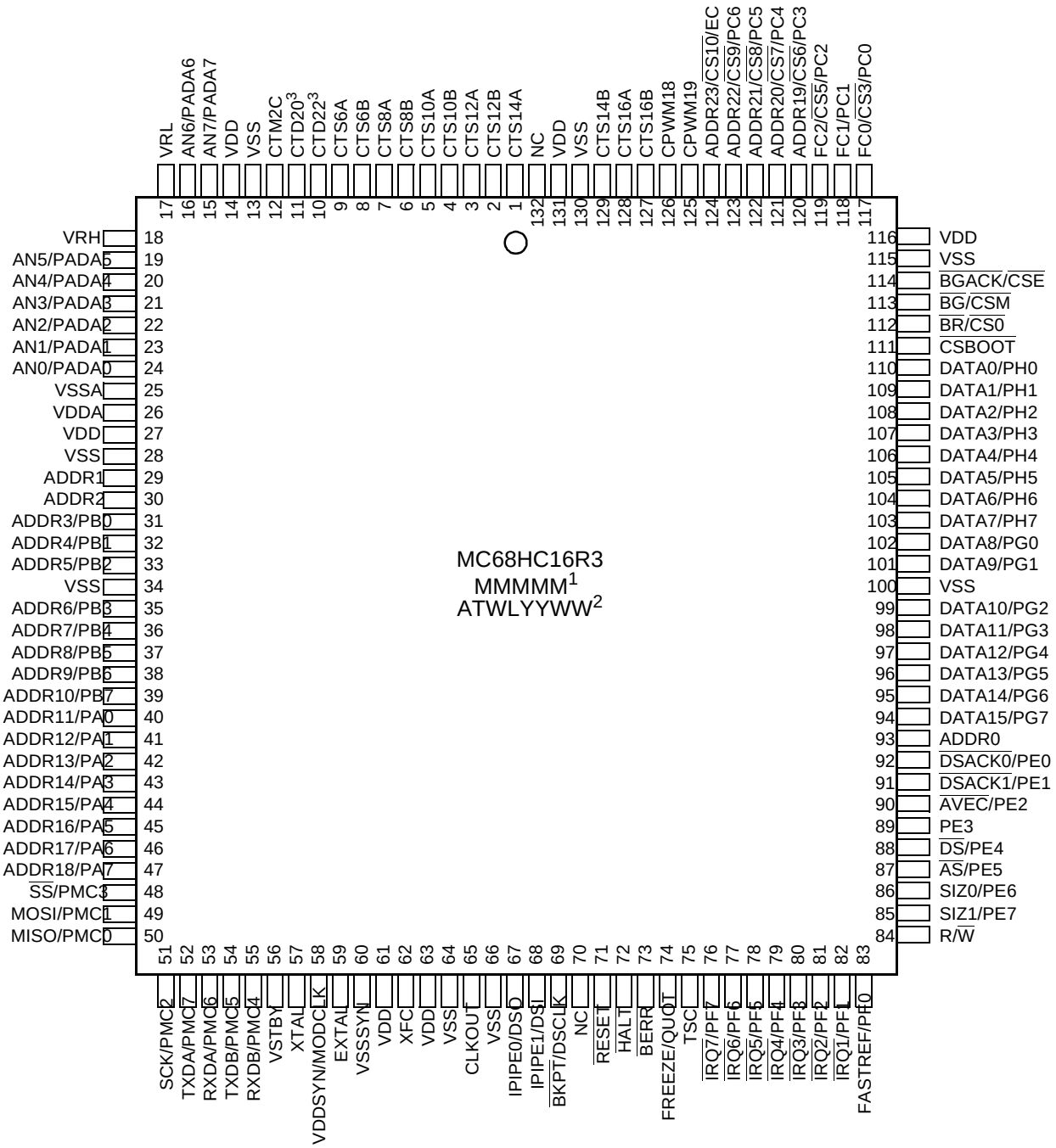


NOTES:

1. MMMMM = MASK OPTION NUMBER
2. ATWLYYWW = ASSEMBLY TEST LOCATION/YEAR, WEEK
3. R-SERIES MCUS ARE COMPATIBLE. THE ONLY DIFFERENCE IS IN THE NAMING OF THE CTM7 AND CTM8 DOUBLE-ACTION SUBMODULE PINS.

MC68HC916R1 132-PIN QF

Figure 3-6 MC68HC916R1 Pin Assignment for 132-Pin Package

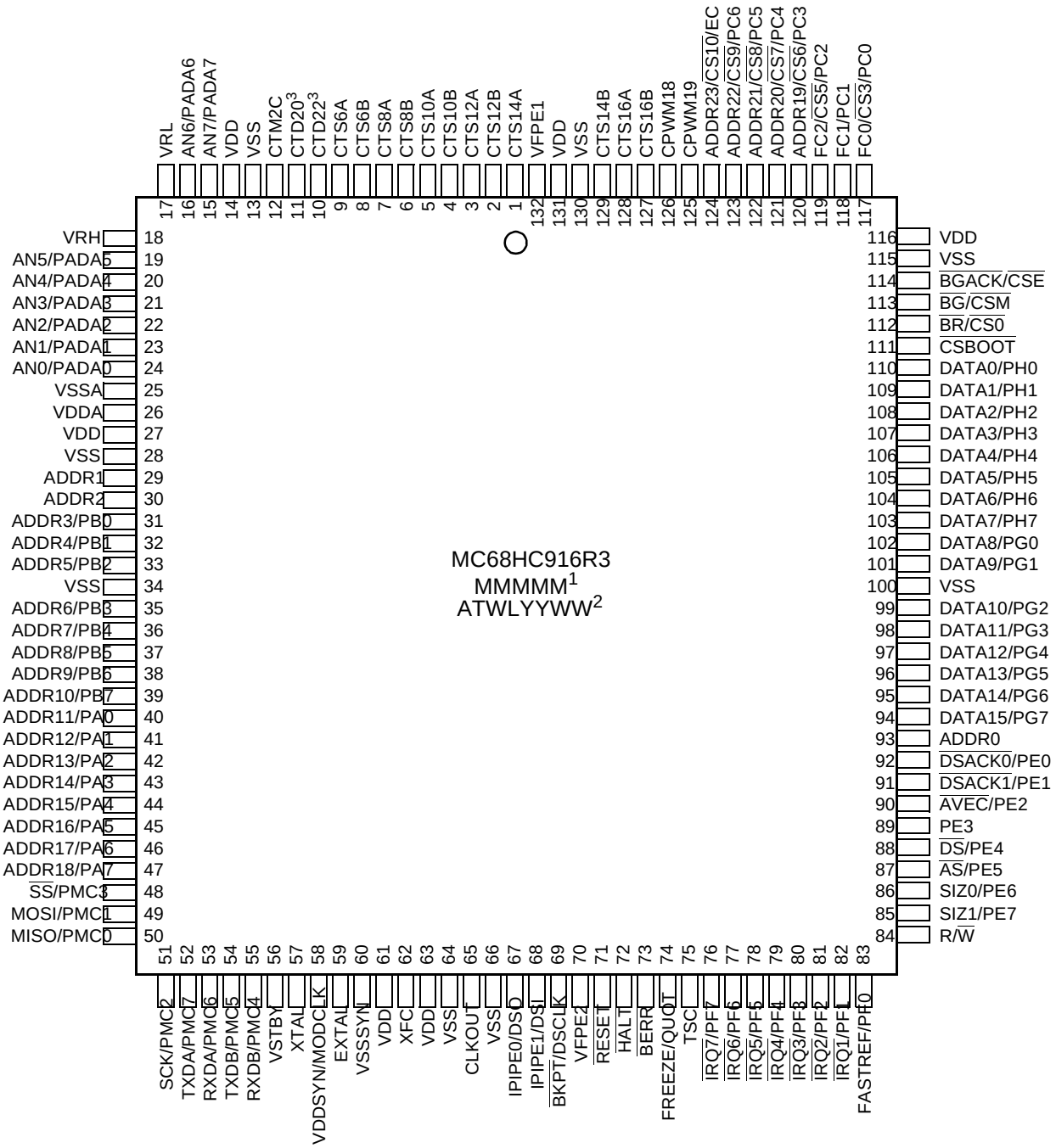


NOTES:

1. MMMMM = MASK OPTION NUMBER
2. ATWLYYWW = ASSEMBLY TEST LOCATION/YEAR, WEEK
3. R-SERIES MCUS ARE COMPATIBLE. THE ONLY DIFFERENCE IS IN THE NAMING OF THE CTM7 AND CTM8 DOUBLE-ACTION SUBMODULE PINS.

MC68HC16R3 132-PIN QF

Figure 3-7 MC68HC16R3 Pin Assignment for 132-Pin Package



NOTES:

1. MMMMM = MASK OPTION NUMBER
2. ATWLYYWW = ASSEMBLY TEST LOCATION/YEAR, WEEK
3. R-SERIES MCUS ARE COMPATIBLE. THE ONLY DIFFERENCE IS IN THE NAMING OF THE CTM7 AND CTM8 DOUBLE-ACTION SUBMODULE PINS.

MC68HC916R3 132-PIN QF

Figure 3-8 MC68HC916R3 Pin Assignment for 132-Pin Package

3.4 Pin Descriptions

Table 3-1 summarizes the pin characteristics of M68HC16 R-series MCUs. Entries in the “Associated Module” column indicate to which module individual pins belong. For MCU pins that can be outputs, the “Driver Type” column lists which output driver type is used. **Table 3-2** briefly describes the four primary driver types. A “—” in the “Driver Type” column indicates either that the pin is an input only, and thus does not have a driver, or that the pin has a special driver, like the XTAL pin. Entries in the “Synchronized Input” and “Input Hysteresis” columns denote whether MCU pins that can be inputs are synchronized to the system clock and if they have hysteresis. Pins that are outputs only or that have special characteristics, like the EXTAL pin, have a “—” in these columns.

Table 3-1 M68HC16 R-Series Pin Characteristics

Pin Mnemonic(s)	Pin Number(s)	Associated Module	Driver Type	Synchronized Input	Input Hysteresis
ADDR0 ADDR1 ADDR2	93 29 30	SCIM2	A	—	—
ADDR3/PB0 ADDR4/PB1 ADDR5/PB2 ADDR6/PB3 ADDR7/PB4 ADDR8/PB5 ADDR9/PB6 ADDR10/PB7	31 32 33 35 36 37 38 39	SCIM2	A	Y	Y
ADDR11/PA0 ADDR12/PA1 ADDR13/PA2 ADDR14/PA3 ADDR15/PA4 ADDR16/PA5 ADDR17/PA6 ADDR18/PA7	40 41 42 43 44 45 46 47	SCIM2	A	Y	Y
ADDR19/$\overline{CS6}$/PC3 ADDR20/$\overline{CS7}$/PC4 ADDR21/$\overline{CS8}$/PC5 ADDR22/$\overline{CS9}$/PC6	120 121 122 123	SCIM2	A	—	—
ADDR23/$\overline{CS10}$/ECLK	124	SCIM2	A	—	—
AN0/PADA0 AN1/PADA1 AN2/PADA2 AN3/PADA3 AN4/PADA4 AN5/PADA5 AN6/PADA6 AN7/PADA7	24 23 22 21 20 19 16 15	ADC	—	Y[1]	Y
$\overline{AS}$/PE5	87	SCIM2	B	Y	Y
$\overline{AVEC}$/PE2	90	SCIM2	B	Y	N
$\overline{BERR}$	73	SCIM2	—	Y[2]	N
$\overline{BG}$/$\overline{CSM}$	113	SCIM2	B	—	—

Table 3-1 M68HC16 R-Series Pin Characteristics (Continued)

Pin Mnemonic(s)	Pin Number(s)	Associated Module	Driver Type	Synchronized Input	Input Hysteresis
$\overline{\text{BGACK}}/\overline{\text{CSE}}$	114	SCIM2	B	Y	N
$\overline{\text{BKPT}}$/DSCLK	69	CPU16	—	Y	Y
$\overline{\text{BR}}/\overline{\text{CS0}}$	112	SCIM2	B	Y	N
CLKOUT	65	SCIM2	A	—	—
CPWM18 CPWM19	126 125	CTM7/CTM8	A	—	—
$\overline{\text{CSBOOT}}$	111	SCIM2	B	—	—
CTD4[3] CTD5[3]	11 10	CTM7	A	Y	Y
CTD20[4] CTD22[4]	11 10	CTM8	A	Y	Y
CTM2C	12	CTM7/CTM8	—	Y	Y
CTS6A CTS6B CTS8A CTS8B CTS10A CTS10B CTS12A CTS12B CTS14A CTS14B CTS16A CTS16B	9 8 7 6 5 4 3 2 1 129 128 127	CTM7/CTM8	A	Y	Y
DATA0/PH0 DATA1/PH1 DATA2/PH2 DATA3/PH3 DATA4/PH4 DATA5/PH5 DATA6/PH6 DATA7/PH7	110 109 108 107 106 105 104 103	SCIM2	Aw	Y[5]	Y
DATA8/PG0 DATA9/PG1 DATA10/PG2 DATA11/PG3 DATA12/PG4 DATA13/PG5 DATA14/PG6 DATA15/PG7	102 101 99 98 97 96 95 94	SCIM2	Aw	Y[5]	Y
$\overline{\text{DS}}$/PE4	88	SCIM2	B	Y	Y
$\overline{\text{DSACK0}}$/PE0 $\overline{\text{DSACK1}}$/PE1	92 91	SCIM2	B	Y	N
EXTAL	59	SCIM2	—	—	—
FASTREF/PF0	83	SCIM2	B	Y[1]	Y
FC0/$\overline{\text{CS3}}$/PC0	117	SCIM2	A	—	—
FC1/PC1	118	SCIM2	A	—	—
FC2/$\overline{\text{CS5}}$/PC2	119	SCIM2	A	—	—

Table 3-1 M68HC16 R-Series Pin Characteristics (Continued)

Pin Mnemonic(s)	Pin Number(s)	Associated Module	Driver Type	Synchronized Input	Input Hysteresis
FREEZE/QUOT	74	SCIM2	A	—	—
$\overline{HALT}$	72	SCIM2	Bo	Y[2]	N
IPIPE0/DSO	67	CPU16	A	—	—
IPIPE1/DSI	68	CPU16	A	Y	Y
$\overline{IRQ1}$/PF1 $\overline{IRQ2}$/PF2 $\overline{IRQ3}$/PF3 $\overline{IRQ4}$/PF4 $\overline{IRQ5}$/PF5 $\overline{IRQ6}$/PF6 $\overline{IRQ7}$/PF7	82 81 80 79 78 77 76	SCIM2	B	Y	Y
MISO/PMC0	50	MCCI	Bo	Y[1]	Y
MOSI/PMC1	49	MCCI	Bo	Y[1]	Y
PE3	89	SCIM2	B	Y	Y
R/$\overline{W}$	84	SCIM2	A	—	—
$\overline{RESET}$	71	SCIM2	Bo	Y	Y
RXDA/PMC6 RXDB/PMC4	53 55	MCCI	Bo	Y[1]	Y
SCK/PMC2	51	MCCI	Bo	Y[1]	Y
SIZ0/PE6 SIZ1/PE7	86 85	SCIM2	B	Y	Y
$\overline{SS}$/PMC3	48	MCCI	Bo	Y[1]	Y
TSC	75	SCIM2	—	Y	Y
TXDA/PMC7 TXDB/PMC5	52 54	MCCI	Bo	Y[1]	Y
V_{DD}	14 27 61 63 116 131	—	—	—	—
V_{DDA}	26	ADC	—	—	—
V_{DDSYN}/MODCLK	58	SCIM2	—	—	—
V_{FPE1}	132	BEFLASH	—	—	—
V_{FPE2}	70	FLASH1 FLASH2	—	—	—
V_{RH} V_{RL}	18 17	ADC	—	—	—
V_{SS}	13 28 34 64 66 100 115 130	—	—	—	—

Table 3-1 M68HC16 R-Series Pin Characteristics (Continued)

Pin Mnemonic(s)	Pin Number(s)	Associated Module	Driver Type	Synchronized Input	Input Hysteresis
V_{SSA}	25	ADC	—	—	—
V_{SSSYN}	60	SCIM2	—	—	—
V_{STBY}	56	SRAM	—	—	—
XFC	62	SCIM2	—	—	—
XTAL	57	SCIM2	—	—	—

NOTES:

1. AN[7:0]/PADA[7:0], FASTREF/PF0, MISO/PMC0, MOSI/PMC1, SCK/PMC2, $\overline{SS}$/PMC3, RXDB/PMC4, TXDB/PMC5, RXDA/PMC6, and TXDA/PMC7 inputs are only synchronized when used as discrete general purpose inputs.
2. $\overline{BERR}$ is only synchronized when executing retry or late bus cycle operations. $\overline{HALT}$ is only synchronized when executing retry or single-step bus cycle operations. These uses of $\overline{HALT}$ and $\overline{BERR}$ are only supported on the CPU32 and not the CPU16.
3. Used only on the MC68HC16R1/916R1.
4. Used only on the MC68HC16R3/916R3.
5. DATA[15:8]/PG[7:0] and DATA[7:0]/PH[7:0] are only synchronized during reset and when being used as discrete general purpose inputs.

Table 3-2 M68HC16 R-Series Driver Types

Type	I/O	Description
A	O	Three-state capable output signals
Aw	O	Type A output with weak p-channel pullup during reset
B	O	Three-state output that includes circuitry to pull up output before high impedance is established, to ensure rapid rise time
Bo	O	Type B output that can be operated in an open-drain mode

Table 3-3 summarizes the pin functions of M68HC16 R-series MCUs. Entries in the "Active State(s)" column denote the polarity of each MCU pin in its active state. Some MCU pins have multiple functions and thus have multiple entries in the "Active State(s)" column. For example, the ADDR23/$\overline{CS10}$/ECLK pin can be programmed to be either address line 23 (ADDR23), chip-select output 10 ($\overline{CS10}$), or the M6800 bus clock (ECLK). Its entry in the "Active State(s)" column is "—/0/—" which indicates the following:

- When programmed as ADDR23, the pin has no active state ("—"); it conveys information when driven by the MCU to logic 0 or logic 1.
- When programmed as $\overline{CS10}$, the pin is active when driven to logic 0 ("0") by the MCU. When driven to logic 1, the chip select function is inactive.
- When programmed as ECLK, the pin has no active state ("—"). M6800 bus devices drive or prepare to latch an address when ECLK is logic 0 and drive or prepare to latch data when ECLK is logic 1.

The "Discrete I/O Use" column indicates whether each pin can be used as a general purpose input, output, or both. Those pins that cannot be used for general purpose I/O will have a "—" in this column.

Table 3-3 M68HC16 R-Series Pin Functions

Pin Mnemonic(s)	Pin Number(s)	Active State(s)	Associated Module	Description	Discrete I/O Use
ADDR0 ADDR1 ADDR2	93 29 30	—	SCIM2	Address lines [2:0].	—
ADDR3/PB0 ADDR4/PB1 ADDR5/PB2 ADDR6/PB3 ADDR7/PB4 ADDR8/PB5 ADDR9/PB6 ADDR10/PB7	31 32 33 35 36 37 38 39	—/—	SCIM2	Address lines [10:3] or digital I/O port B [7:0].	I/O
ADDR11/PA0 ADDR12/PA1 ADDR13/PA2 ADDR14/PA3 ADDR15/PA4 ADDR16/PA5 ADDR17/PA6 ADDR18/PA7	40 41 42 43 44 45 46 47	—/—	SCIM2	Address lines [18:11] or digital I/O port A [7:0].	I/O
ADDR19/$\overline{CS6}$/PC3 ADDR20/$\overline{CS7}$/PC4 ADDR21/$\overline{CS8}$/PC5 ADDR22/$\overline{CS9}$/PC6	120 121 122 123	—/0/—	SCIM2	Address lines [22:19], chip select outputs [9:6], or digital output port C [6:3].	O
ADDR23/$\overline{CS10}$/ECLK	124	—/0/—	SCIM2	Address line 23, chip select output 10, or E clock output for M6800 bus devices.	—
AN0/PADA0 AN1/PADA1 AN2/PADA2 AN3/PADA3 AN4/PADA4 AN5/PADA5 AN6/PADA6 AN7/PADA7	24 23 22 21 20 19 16 15	—/—	ADC	Analog inputs to ADC multiplexer or digital input port ADA [7:0].	I
$\overline{AS}$/PE5	87	0/—	SCIM2	Indicates that a valid address is on the address bus or digital I/O port E5.	I/O
$\overline{AVEC}$/PE2	90	0/—	SCIM2	Requests an automatic vector during an interrupt acknowledge cycle or digital I/O port E2.	I/O
$\overline{BERR}$	73	0	SCIM2	Requests a bus error exception.	—
$\overline{BG}$/$\overline{CSM}$	113	0/0	SCIM2	Bus granted output or emulation memory chip select output.	—
$\overline{BGACK}$/$\overline{CSE}$	114	0/0	SCIM2	Bus grant acknowledge input or SCIM2 emulation chip select output.	—
$\overline{BKPT}$/DSCLK	69	0/—	CPU16	Hardware breakpoint input or background debug mode serial data clock input.	—
$\overline{BR}$/$\overline{CS0}$	112	0/0	SCIM2	Bus request input or chip select output 0.	—
CLKOUT	65	—	SCIM2	System clock output.	—

Table 3-3 M68HC16 R-Series Pin Functions (Continued)

Pin Mnemonic(s)	Pin Number(s)	Active State(s)	Associated Module	Description	Discrete I/O Use
CPWM18 CPWM19	126 125	—	CTM7/CTM8	Pulse width modulation submodule (PWMSM) outputs or digital output ports.	O[1]
$\overline{\text{CSBOOT}}$	111	0	SCIM2	Boot memory device chip select output.	—
CTD4[2] CTD5[2]	11 10	—	CTM7	Bidirectional double action submodule (DASM) timer pins or digital I/O ports.	I/O[1]
CTD20[3] CTD22[3]	11 10	—	CTM8	Bidirectional double action submodule (DASM) timer pins or digital I/O ports.	I/O[1]
CTM2C	12	—	CTM7/CTM8	Free-running counter submodule (FCSM) and modulus counter submodule (MCSM) external clock input or digital input port.	I[1]
CTS6A CTS6B CTS8A CTS8B CTS10A CTS10B CTS12A CTS12B CTS14A CTS14B CTS16A CTS16B	9 8 7 6 5 4 3 2 1 129 128 127	—	CTM7/CTM8	Bidirectional single action submodule (SASM) timer pins or digital I/O ports.	I/O
DATA0/PH0 DATA1/PH1 DATA2/PH2 DATA3/PH3 DATA4/PH4 DATA5/PH5 DATA6/PH6 DATA7/PH7	110 109 108 107 106 105 104 103	—/—	SCIM2	Data bus lines [7:0] or digital I/O port H [7:0].	I/O
DATA8/PG0 DATA9/PG1 DATA10/PG2 DATA11/PG3 DATA12/PG4 DATA13/PG5 DATA14/PG6 DATA15/PG7	102 101 99 98 97 96 95 94	—/—	SCIM2	Data bus lines [15:8] or digital I/O port G [7:0.]	I/O
$\overline{\text{DS}}$/PE4	88	0/—	SCIM2	Indicates that an external device should place valid data on the bus during a read cycle, that valid data has been placed on the bus during a write cycle, or the pin is digital I/O port E4.	I/O
$\overline{\text{DSACK0}}$/PE0 $\overline{\text{DSACK1}}$/PE1	92 91	0/—	SCIM2	Data size and acknowledge inputs or digital I/O ports E [1:0].	I/O

Table 3-3 M68HC16 R-Series Pin Functions (Continued)

Pin Mnemonic(s)	Pin Number(s)	Active State(s)	Associated Module	Description	Discrete I/O Use
EXTAL	59	—	SCIM2	Crystal oscillator or external clock input.	—
FASTREF/PF0	83	1/—	SCIM2	Phase-locked loop reference select input or digital I/O port F0.	I/O
FC0/$\overline{CS3}$/PC0	117	—/0/—	SCIM2	Function code output 0, chip select output 3, or digital output port C0.	O
FC1/PC1	118	—/—	SCIM2	Function code output 1 or digital output port C1.	O
FC2/$\overline{CS5}$/PC2	119	—/0/—	SCIM2	Function code output 2, chip select output 5, or digital output port C2.	O
FREEZE/QUOT	74	1/—	SCIM2	Indicates that the CPU16 has entered background debug mode or provides the quotient bit of the polynomial divider in test mode.	—
$\overline{HALT}$	72	0	SCIM2	Suspends bus activity.	—
IPIPE0/DSO	67	—/—	CPU16	Instruction pipeline state output 0 or background debug mode serial data output.	—
IPIPE1/DSI	68	—/—	CPU16	Instruction pipeline state output 1 or background debug mode serial data input.	—
$\overline{IRQ1}$/PF1 $\overline{IRQ2}$/PF2 $\overline{IRQ3}$/PF3 $\overline{IRQ4}$/PF4 $\overline{IRQ5}$/PF5 $\overline{IRQ6}$/PF6 $\overline{IRQ7}$/PF7	82 81 80 79 78 77 76	0/—	SCIM2	External interrupt request inputs [7:1] or digital I/O port F [7:1].	I/O
MISO/PMC0	50	—/—	MCCI	SPI master input/slave output data or digital I/O port MC0.	I/O
MOSI/PMC1	49	—/—	MCCI	SPI master output/slave input data or digital I/O port MC1.	I/O
PE3	89	—	SCIM2	Digital I/O port E3.	I/O
R/$\overline{W}$	84	1/0	SCIM2	Indicates a data bus read when high and a data bus write when low.	—
$\overline{RESET}$	71	0	SCIM2	System reset.	—
RXDA/PMC6 RXDB/PMC4	53 55	—/—	MCCI	SCI A and B receive data inputs or digital I/O ports MC6 and MC4	I/O
SCK/PMC2	51	—/—	MCCI	SPI serial clock input/output or digital I/O port MC2.	I/O
SIZ0/PE6 SIZ1/PE7	86 85	—/—	SCIM2	Data transfer size outputs or digital I/O ports E [7:6].	I/O
$\overline{SS}$/PMC3	48	0/—	MCCI	SPI slave select input or digital I/O port MC3.	I/O
TSC	75	1	SCIM2	Places MCU outputs in high-impedance state.	—
TXDA/PMC7 TXDB/PMC5	52 54	—/—	MCCI	SCI A and B transmit data outputs or digital I/O ports MC7 and MC5.	I/O

Table 3-3 M68HC16 R-Series Pin Functions (Continued)

Pin Mnemonic(s)	Pin Number(s)	Active State(s)	Associated Module	Description	Discrete I/O Use
V_{DD}	14 27 61 63 116 131	—	—	Digital supply voltage inputs.	—
V_{DDA}	26	—	ADC	ADC analog supply voltage input.	—
V_{DDSYN}/MODCLK	58	—/1	SCIM2	Clock synthesizer power supply input. If V_{DDSYN} is grounded, the MCU will operate at the frequency of the signal input on the EXTAL pin.	—
V_{FPE1}	132	—	BEFLASH	Block-erasable flash EEPROM program/erase supply voltage input.	—
V_{FPE2}	70	—	FLASH1 FLASH2	Flash EEPROM program/erase supply voltage inputs.	—
V_{RH} V_{RL}	18 17	—	ADC	Analog-to-digital converter high and low voltage reference inputs.	—
V_{SS}	13 28 34 64 66 100 115 130	—	—	Digital ground reference.	—
V_{SSA}	25	—	ADC	ADC analog ground reference.	—
V_{SSSYN}	60	—	SCIM2	Clock synthesizer ground reference.	—
V_{STBY}	56	—	SRAM	SRAM standby voltage supply input.	—
XFC	62	—	SCIM2	Clock synthesizer filter connection.	—
XTAL	57	—	SCIM2	Crystal oscillator output.	—

NOTES:
1. CTM7/CTM8 pins that can be used for general purpose I/O are not grouped into ports like PORTMC on the MCCI and PORTADA on the ADC. Instead, the I/O capability of the pin(s) associated with each CTM7/CTM8 submodule is controlled individually by bits in the submodule's status/interrupt/control register.
2. Used only on the MC68HC16R1/916R1.
3. Used only on the MC68HC16R3/916R3.

3.5 CPU16 Memory Mapping

Each member of the M68HC16 family is comprised of a set of modules connected by the intermodule bus (IMB). The full IMB has a 16-bit data bus, a 24-bit address bus, and three function code lines, and ideally provides eight distinct memory maps, each with 16 megabytes of address space. In practice, only four of these memory maps are available for user code and data. Three are inaccessible because the function codes lines are never driven to states that allow them to be decoded, and one is devoted exclusively to control information not associated with normal read and write bus cycles.

The total amount of addressable memory is further limited on the CPU16. While the CPU32 can operate in both the user and supervisor modes denoted by the function code lines, the CPU16 operates only in supervisor mode. Excluding the CPU space memory map used for special bus cycles, the CPU16 can access only the supervisor program space and supervisor data space memory maps. The CPU16 also has only 20 address lines. This limits the total address space in each of the two memory maps to one megabyte.

Although the CPU16 has only 20 address lines, it still drives all 24 IMB address lines. IMB address lines [19:0] follow CPU address lines [19:0], and IMB address lines [23:20] follow the state of CPU address line 19 as shown in **Figure 3-9**. This causes an address space discontinuity to appear on the IMB when the CPU16 address bus rolls over from $7FFFF to $80000.

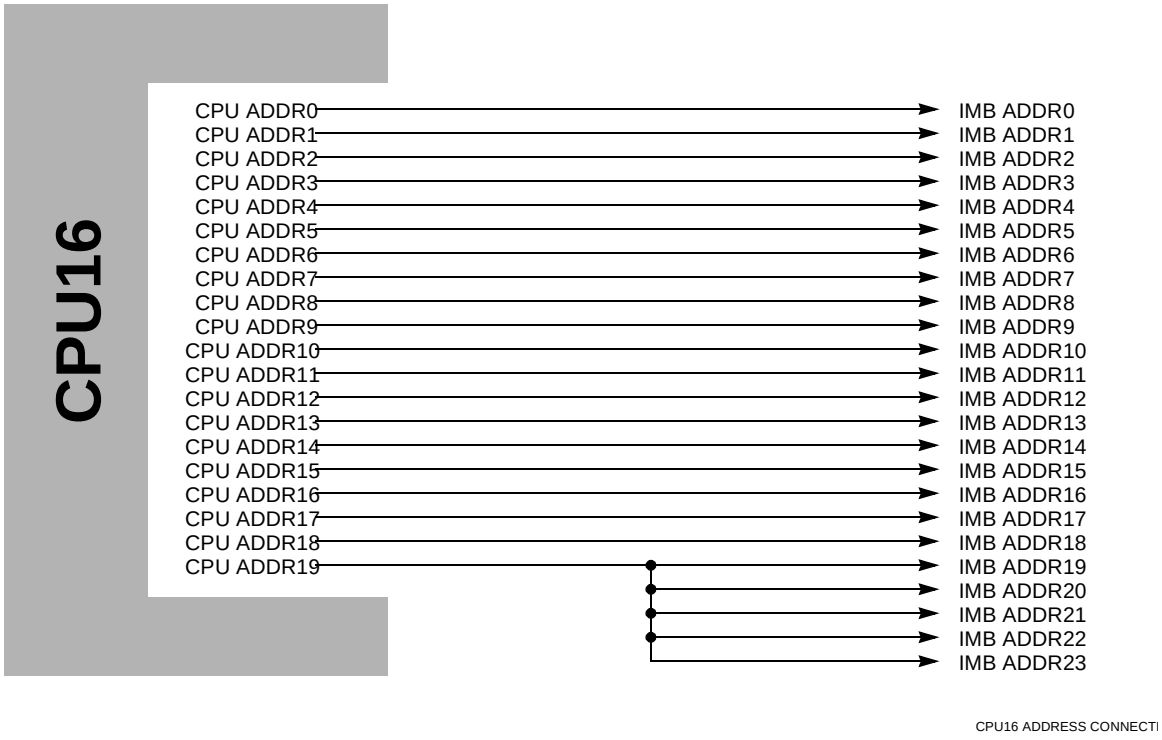


Figure 3-9 Address Bus Connections Between the CPU16 and IMB

Each address space boundary condition is outlined by the statements that follow. Consider **Figure 3-9** and the relationship between CPU address line 19 and IMB address lines [23:20] when examining these boundary conditions. The first boundary condition occurs when the CPU16 drives $7FFFF onto its address bus and is derived as follows.

1. If CPU ADDR[19:0] = $7FFFF = %0111 1111 1111 1111 1111
2. Then CPU ADDR19 = %0 and IMB ADDR19 = %0
3. Consequently, IMB ADDR[23:20] = %0000 = $0
4. Thus IMB ADDR[23:0] = $07FFFF = %0000 0111 1111 1111 1111 1111

The second boundary condition occurs when the CPU16 drives $80000 onto its address bus and is derived as follows.

1. If CPU ADDR[19:0] = $80000 = %1000 0000 0000 0000 0000,
2. Then CPU ADDR19 = %1 and IMB ADDR19 = %1
3. Consequently, IMB ADDR[23:20] = %1111 = $F,
4. Thus IMB ADDR[23:0] = $F80000 = %1111 1000 0000 0000 0000 0000

As the above boundary conditions illustrate, addresses between $080000 and $F7FFFF will never be seen on the IMB of a CPU16 derivative. At no time will IMB address lines [23:20] be driven to states opposite that of CPU address line 19.

It is important to note that this gap is present on the IMB only. The CPU16 simply sees a flat one megabyte memory map from $00000 to $FFFFF, and user software need only generate 20-bit effective addresses to access any location in this range.

3.6 Internal Register Maps

In **Figures 3-10** through **3-13**, IMB address lines [23:20] are represented by the letter Y. The value of Y is equal to %M111, where M is the logic state of the module mapping (MM) bit in the single-chip integration module configuration register (SCIMCR).

NOTE

MM must remain set to logic 1 on all CPU16 derivatives in order for MCU control registers to remain accessible.

As discussed in **3.5 CPU16 Memory Mapping**, CPU16 address lines [19:0] drive IMB address lines [19:0] and CPU16 address line 19 drives IMB address lines [23:20]. For this reason, addresses between $080000 and $F7FFFF will never be seen on the IMB. Setting MM to logic 0 on M68HC16 R-series MCUs would map the control registers from $7FF700 to $7FFC3F where they would be inaccessible until a reset occurs.

As long as MM is set to logic 1, MCU control registers will be accessible, and the CPU16 need only generate 20-bit effective addresses to access them. Thus to access SCIMCR, which is mapped at IMB address $YFFA00, the CPU16 must generate the 20-bit effective address $FFA00.

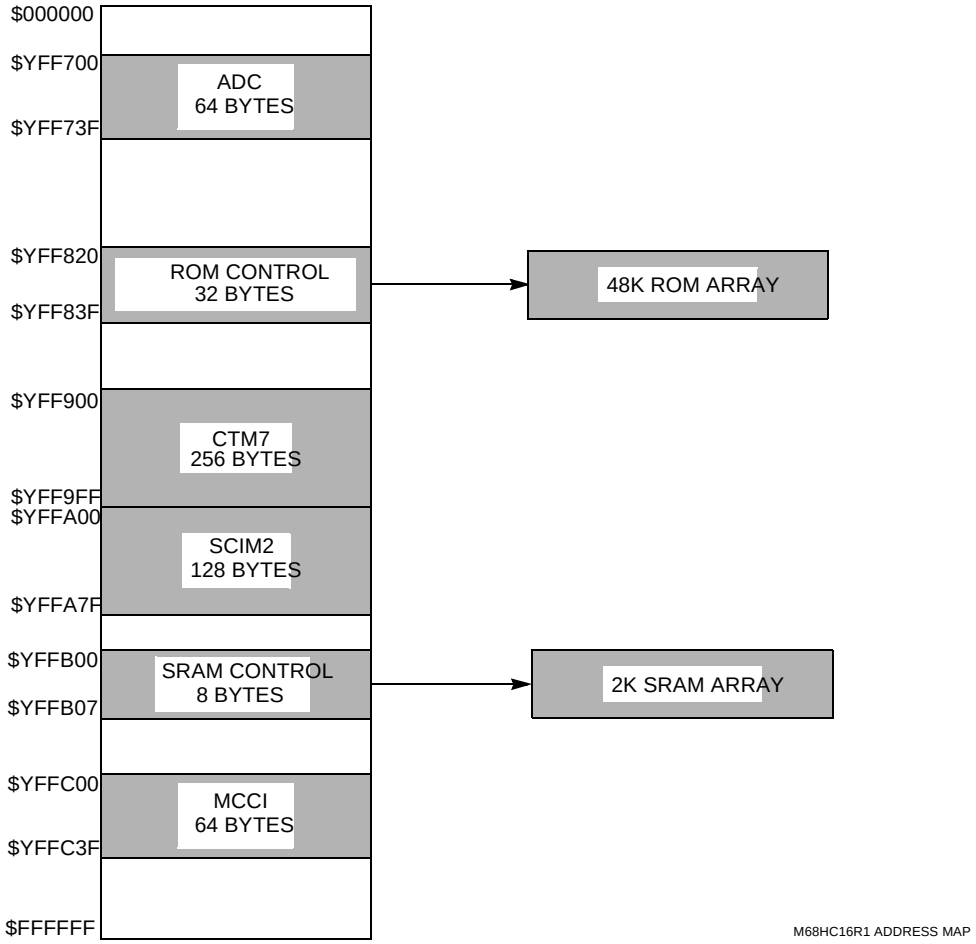


Figure 3-10 MC68HC16 R1 Address Map

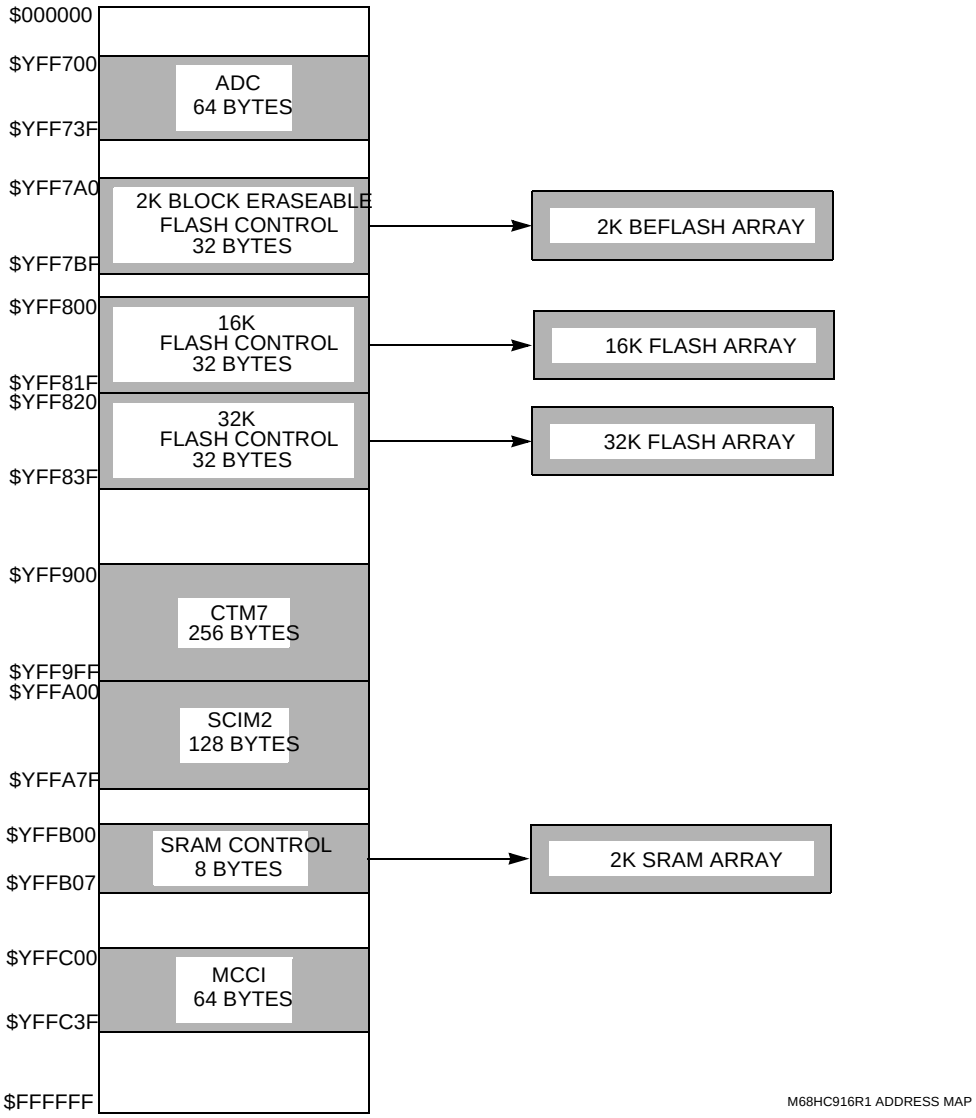


Figure 3-11 MC68HC916R1 Address Map

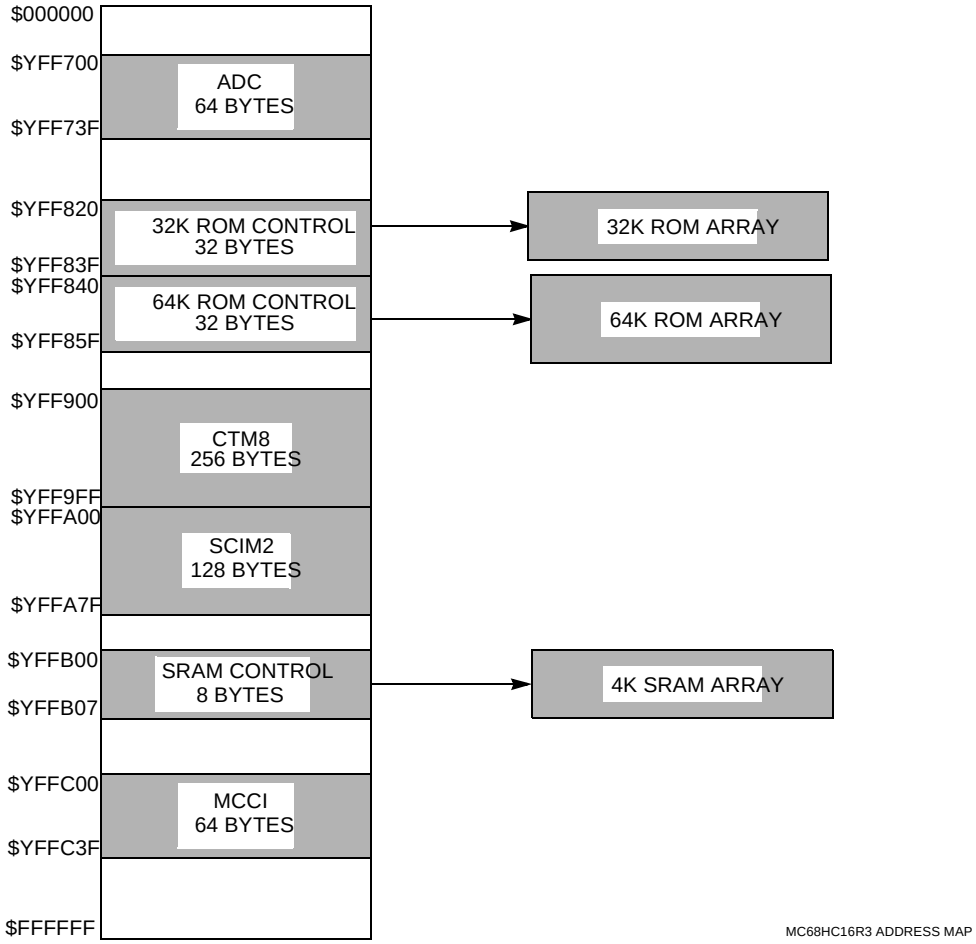


Figure 3-12 MC68HC16 R3 Address Map

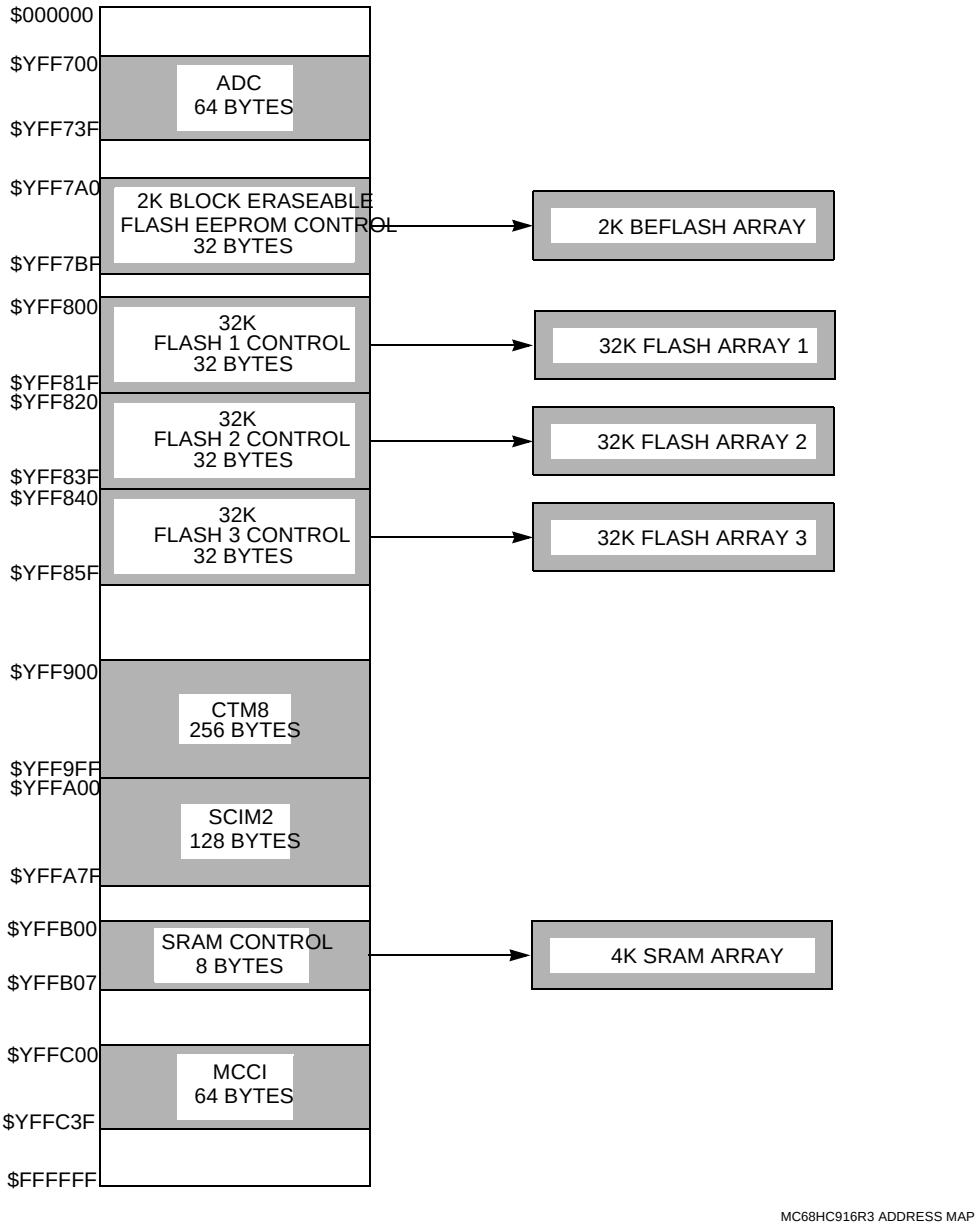
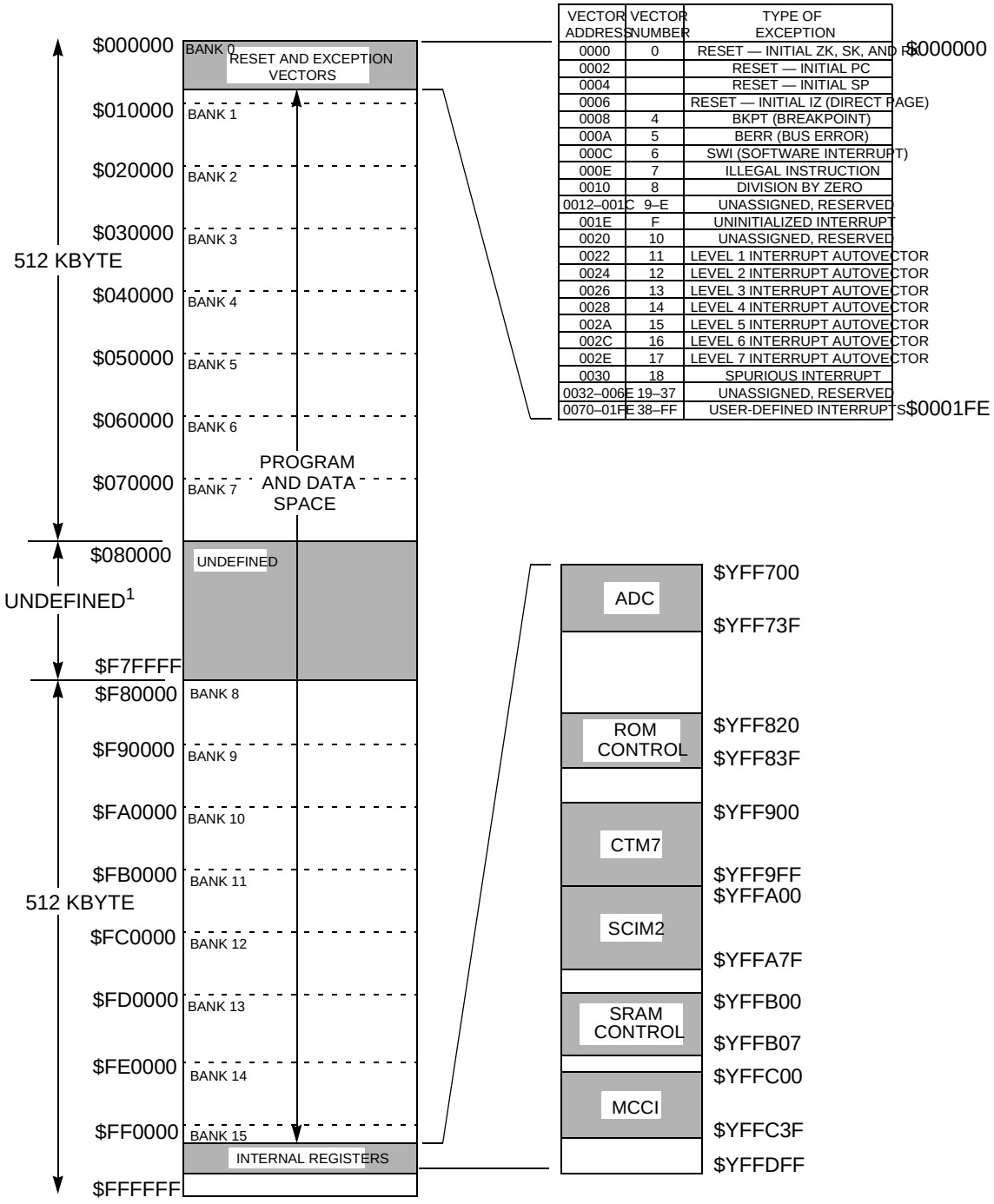


Figure 3-13 MC68HC916R3 Address Map

3.7 Address Space Maps

Figures 3-16 and **3-17** show CPU16 address space for the MC68HC16R1 MCU. **Figures 3-16** and **3-17** show CPU16 address space for the MC68HC916R1 MCU. Address space can be split into physically distinct program and data spaces by decoding the MCU function code outputs. **Figures 3-16** and **3-16** show the memory map of a system that has combined program and data spaces. **Figures 3-17** and **3-17** show the memory map when MCU function code outputs are decoded.

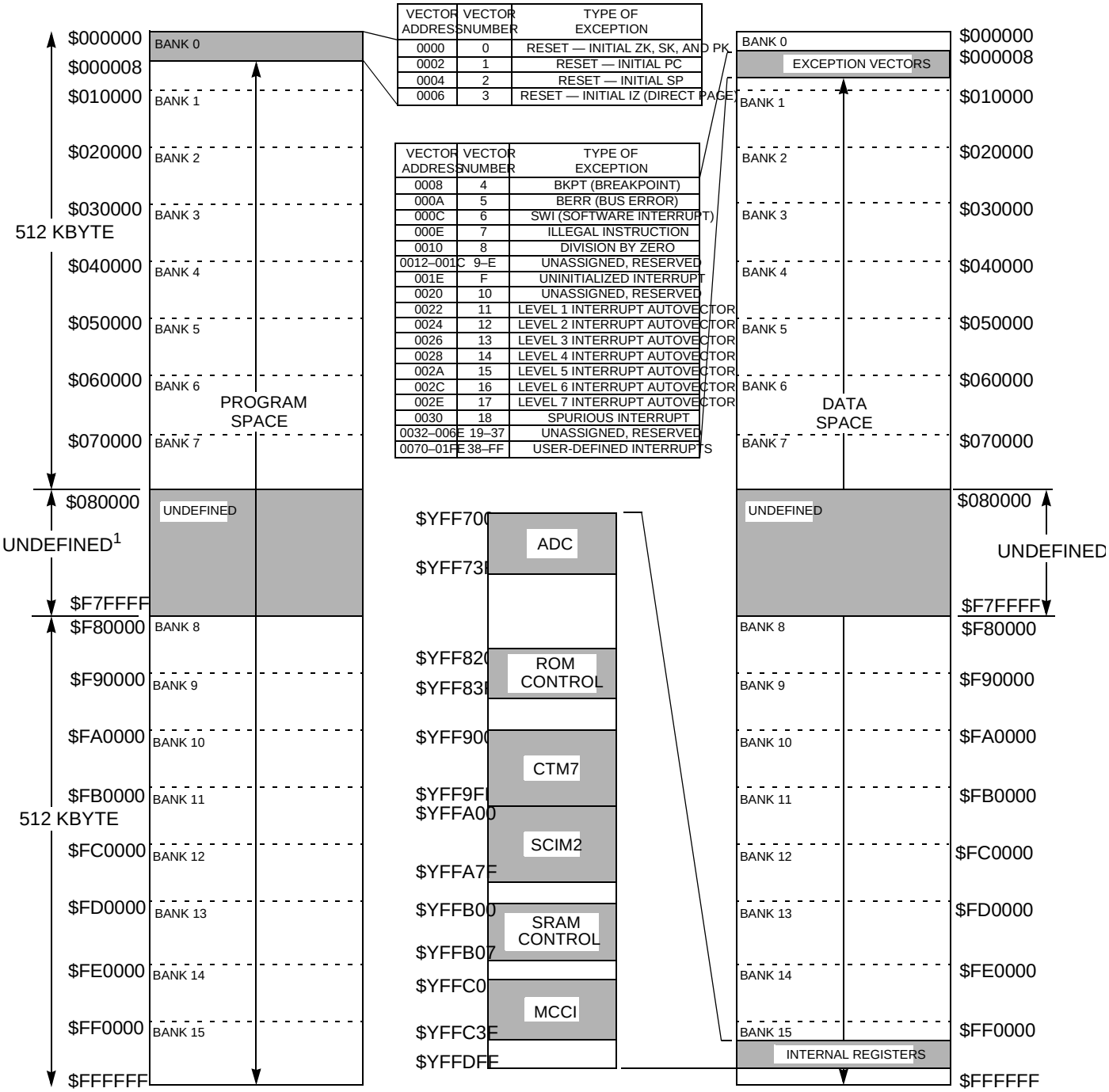
Reset and exception vectors are mapped into bank 0 and cannot be relocated. The CPU16 program counter, stack pointer, and Z index register can be initialized to any address in memory, but exception vectors are limited to 16-bit addresses. To access locations outside of bank 0 during exception handler routines (including interrupt exceptions), a jump table must be used. Refer to SECTION 4 CENTRAL PROCESSOR UNIT for more information on extended addressing and exception processing. Refer to SECTION 5 SINGLE-CHIP INTEGRATION MODULE 2 for more information concerning function codes, address space types, resets, and interrupts.



NOTE:

1. THE ADDRESSES DISPLAYED IN THIS MEMORY MAP ARE THE FULL 24- BIT IMB ADDRESSES. THE CPU16 ADDRESS BUS IS 20 BITS WIDE, AND CPU16 ADDRESS LINE 19 DRIVES IMB ADDRESS LINES [23:20]. THE BLOCK OF ADDRESSES FROM $080000 TO $F7FFFF MARKED AS UNDEFINED WILL NEVER APPEAR ON THE IMB. MEMORY BANKS 0 TO 15 APPEAR FULLY CONTIGUOUS IN THE CPU16'S FLAT 20-BIT ADDRESS SPACE. THE CPU16 NEED ONLY GENERATE A 20-BIT EFFECTIVE ADDRESS TO ACCESS ANY LOCATION IN THIS RANGE.

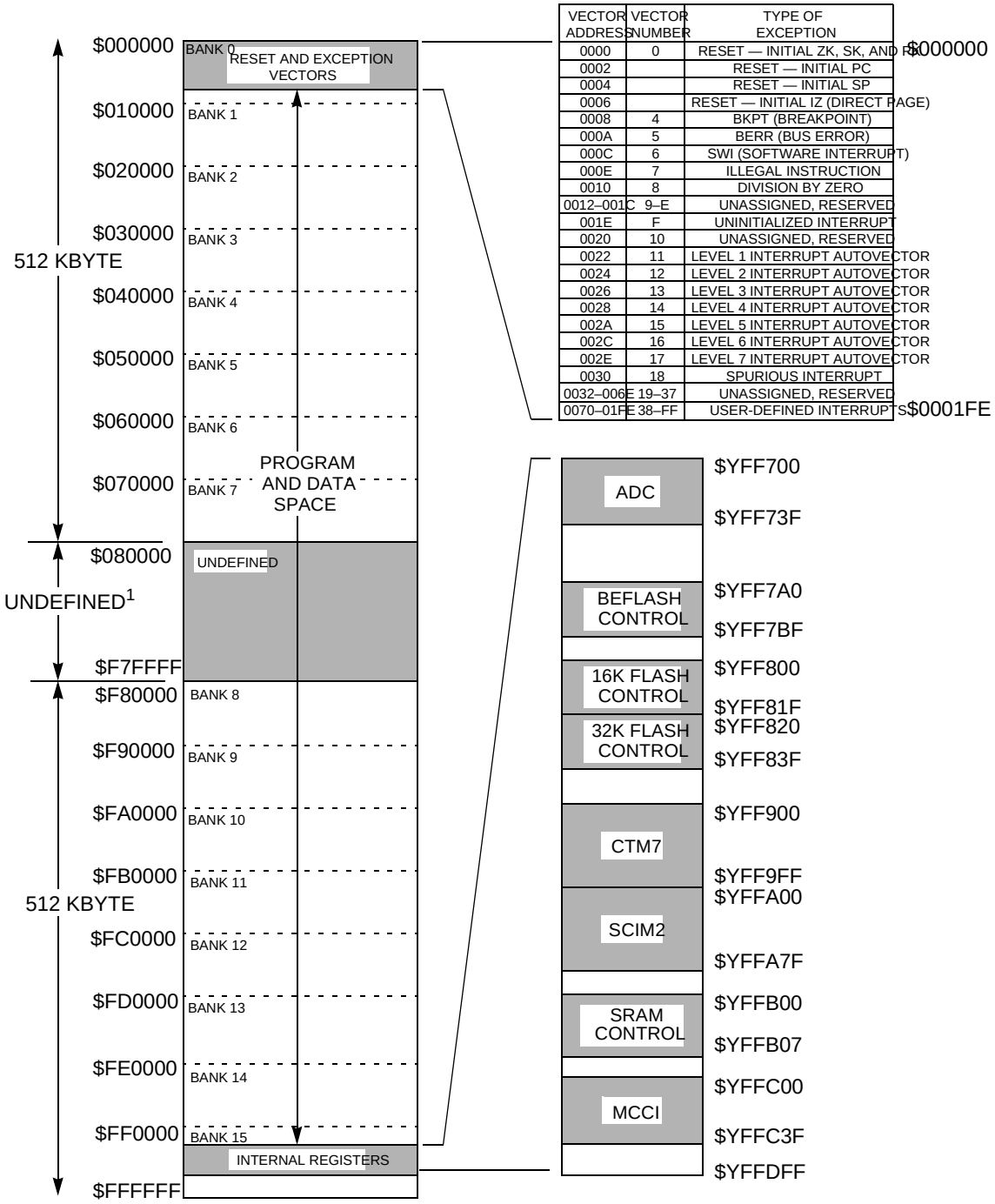
Figure 3-14 MC68HC16R1 Combined Program and Data Space Map



NOTE:
1. THE ADDRESSES DISPLAYED IN THIS MEMORY MAP ARE THE FULL 24- BIT IMB ADDRESSES. THE CPU16 ADDRESS BUS IS 20 BITS WIDE, AND CPU16 ADDRESS LINE 19 DRIVES IMB ADDRESS LINES [23:20]. THE BLOCK OF ADDRESSES FROM $080000 TO $F7FFFF MARKED AS UNDEFINED WILL NEVER APPEAR ON THE IMB. MEMORY BANKS 0 TO 15 APPEAR FULLY CONTIGUOUS IN THE CPU16'S FLAT 20-BIT ADDRESS SPACE. THE CPU16 NEED ONLY GENERATE A 20-BIT EFFECTIVE ADDRESS TO ACCESS ANY LOCATION IN THIS RANGE.

16R1 MEM MAP (S)

Figure 3-15 MC68HC16R1 Separate Program and Data Space Map

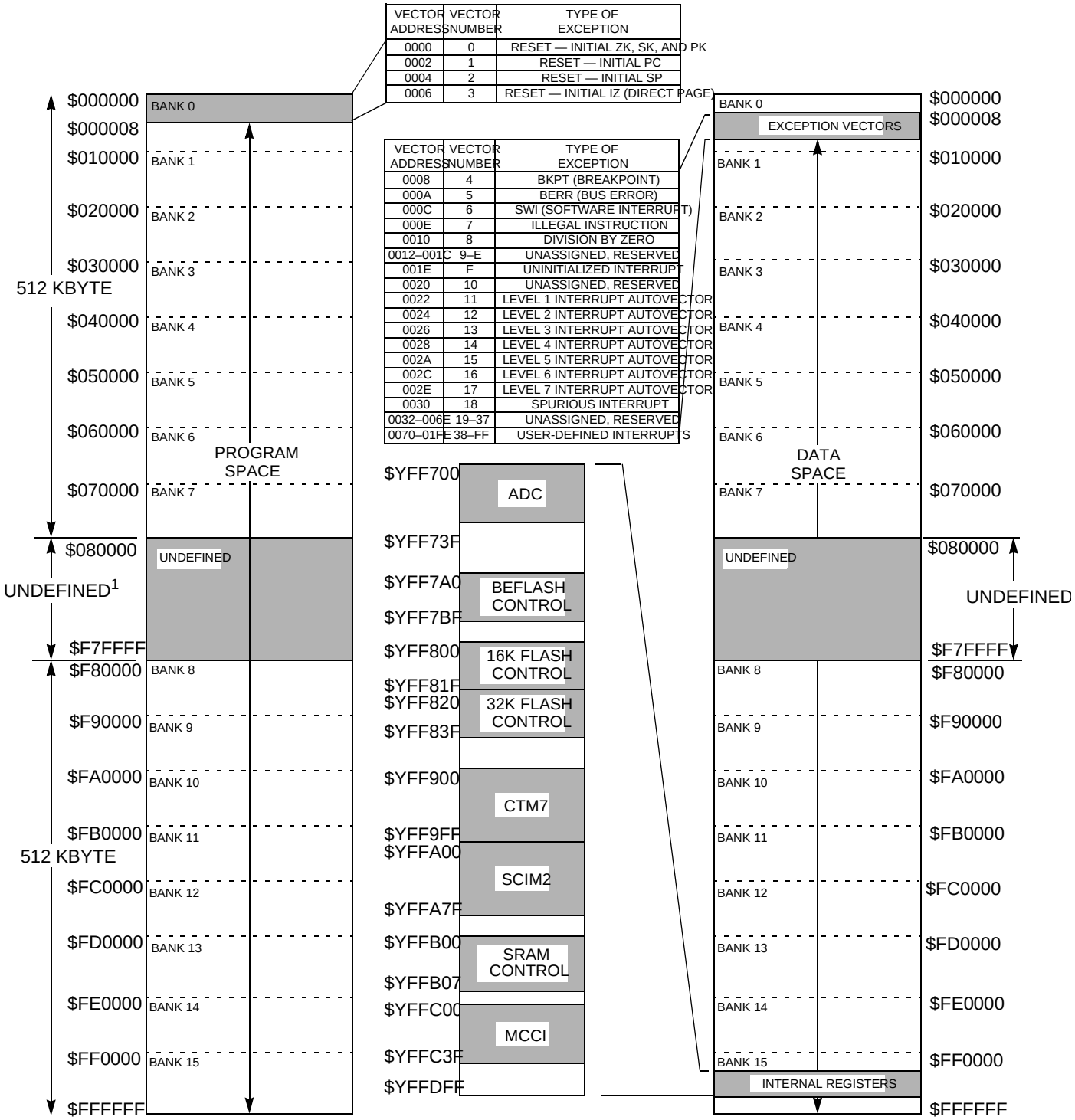


NOTE:

1. THE ADDRESSES DISPLAYED IN THIS MEMORY MAP ARE THE FULL 24- BIT IMB ADDRESSES. THE CPU16 ADDRESS BUS IS 20 BITS WIDE, AND CPU16 ADDRESS LINE 19 DRIVES IMB ADDRESS LINES [23:20]. THE BLOCK OF ADDRESSES FROM $080000 TO $F7FFFF MARKED AS UNDEFINED WILL NEVER APPEAR ON THE IMB. MEMORY BANKS 0 TO 15 APPEAR FULLY CONTIGUOUS IN THE CPU16'S FLAT 20-BIT ADDRESS SPACE. THE CPU16 NEED ONLY GENERATE A 20-BIT EFFECTIVE ADDRESS TO ACCESS ANY LOCATION IN THIS RANGE.

916R1 MEM MAP (C)

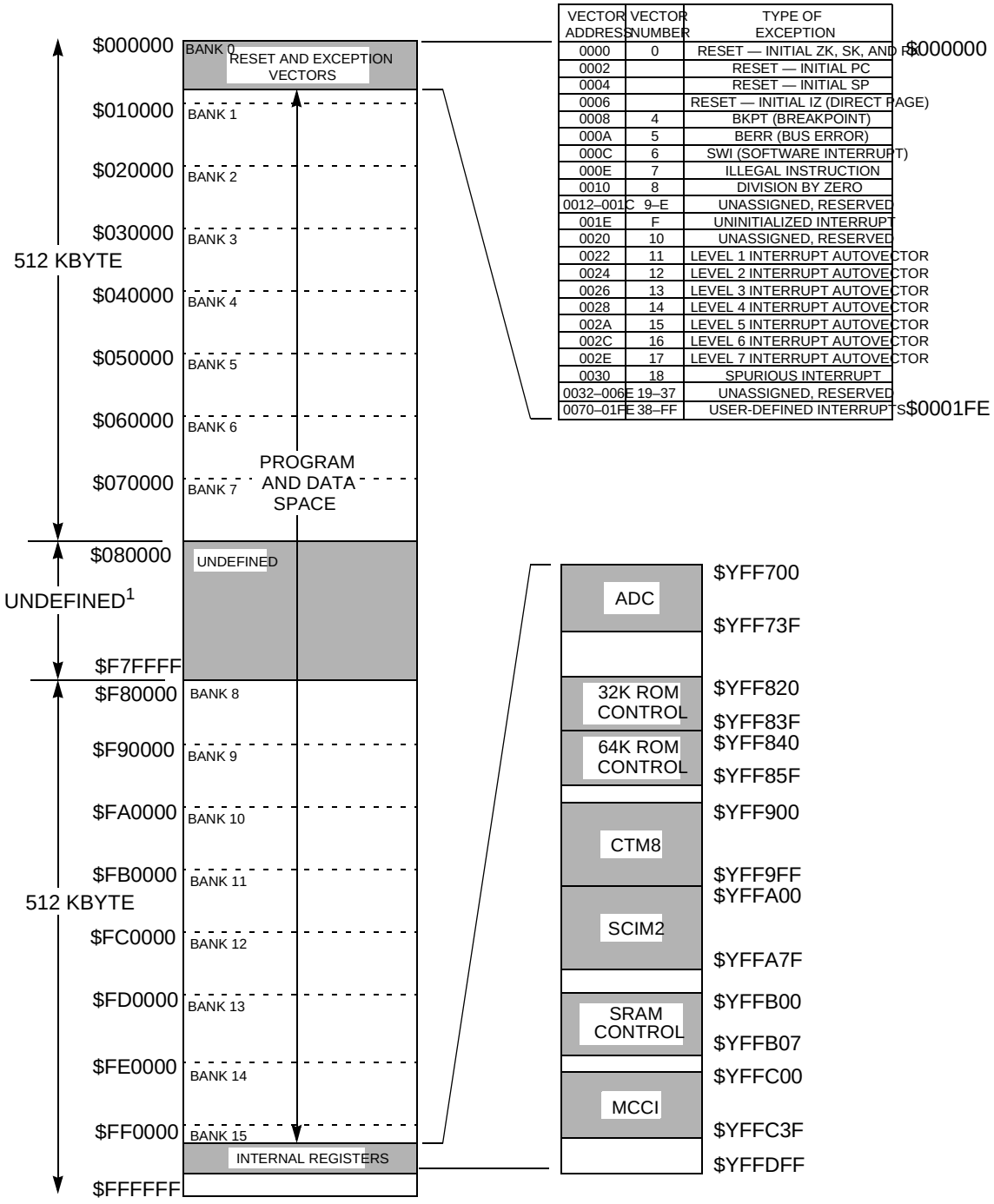
Figure 3-16 MC68HC916R1 Combined Program and Data Space Map



NOTE:
1. THE ADDRESSES DISPLAYED IN THIS MEMORY MAP ARE THE FULL 24- BIT IMB ADDRESSES. THE CPU16 ADDRESS BUS IS 20 BITS WIDE, AND CPU16 ADDRESS LINE 19 DRIVES IMB ADDRESS LINES [23:20]. THE BLOCK OF ADDRESSES FROM $080000 TO $F7FFFF MARKED AS UNDEFINED WILL NEVER APPEAR ON THE IMB. MEMORY BANKS 0 TO 15 APPEAR FULLY CONTIGUOUS IN THE CPU16'S FLAT 20-BIT ADDRESS SPACE. THE CPU16 NEED ONLY GENERATE A 20-BIT EFFECTIVE ADDRESS TO ACCESS ANY LOCATION IN THIS RANGE.

916R1 MEM MAP (S)

Figure 3-17 MC68HC916R1 Separate Program and Data Space Map

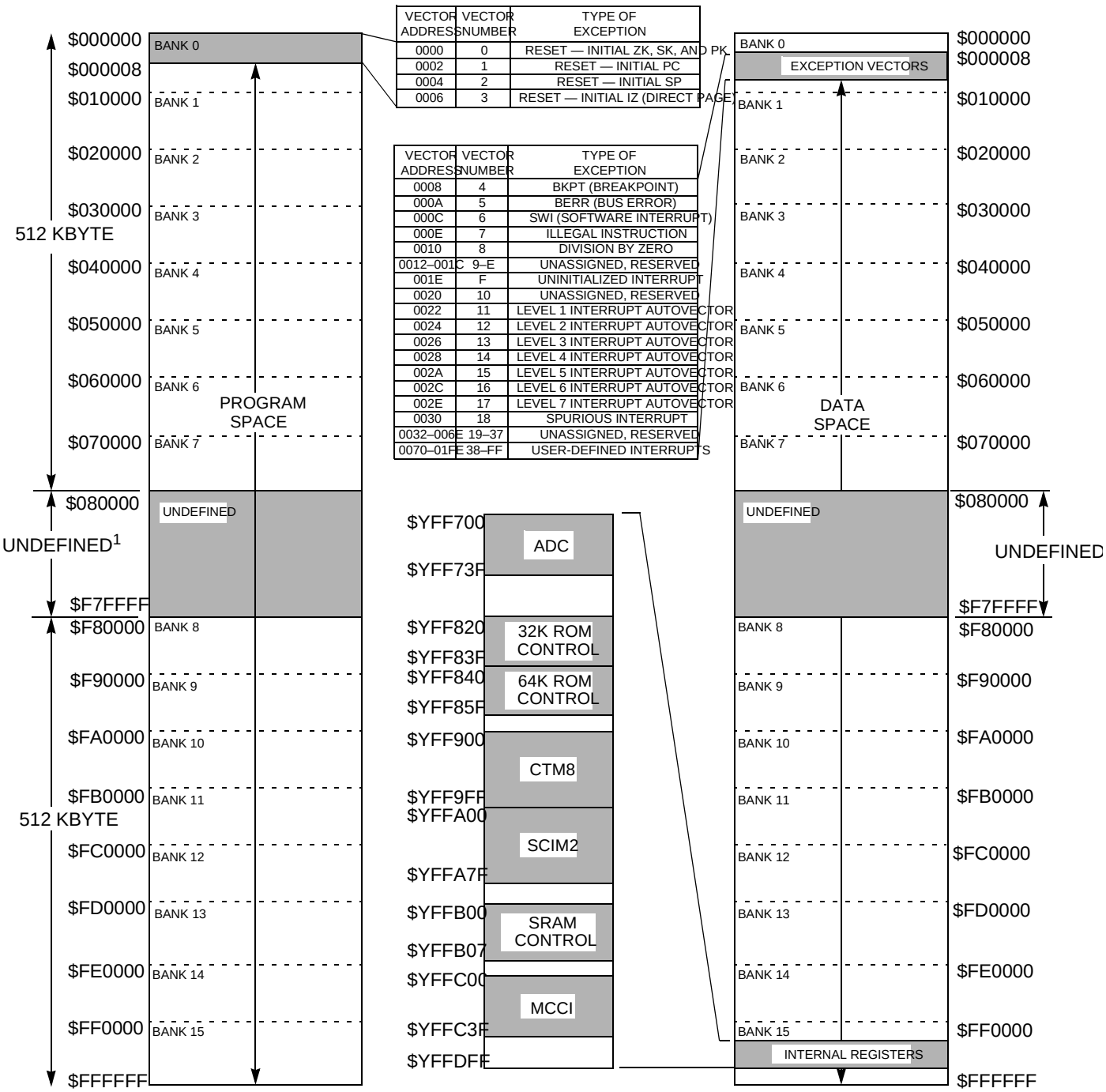


NOTE:

1. THE ADDRESSES DISPLAYED IN THIS MEMORY MAP ARE THE FULL 24- BIT IMB ADDRESSES. THE CPU16 ADDRESS BUS IS 20 BITS WIDE, AND CPU16 ADDRESS LINE 19 DRIVES IMB ADDRESS LINES [23:20]. THE BLOCK OF ADDRESSES FROM $080000 TO $F7FFFF MARKED AS UNDEFINED WILL NEVER APPEAR ON THE IMB. MEMORY BANKS 0 TO 15 APPEAR FULLY CONTIGUOUS IN THE CPU16'S FLAT 20-BIT ADDRESS SPACE. THE CPU16 NEED ONLY GENERATE A 20-BIT EFFECTIVE ADDRESS TO ACCESS ANY LOCATION IN THIS RANGE.

16R3 MEM MAP (C)

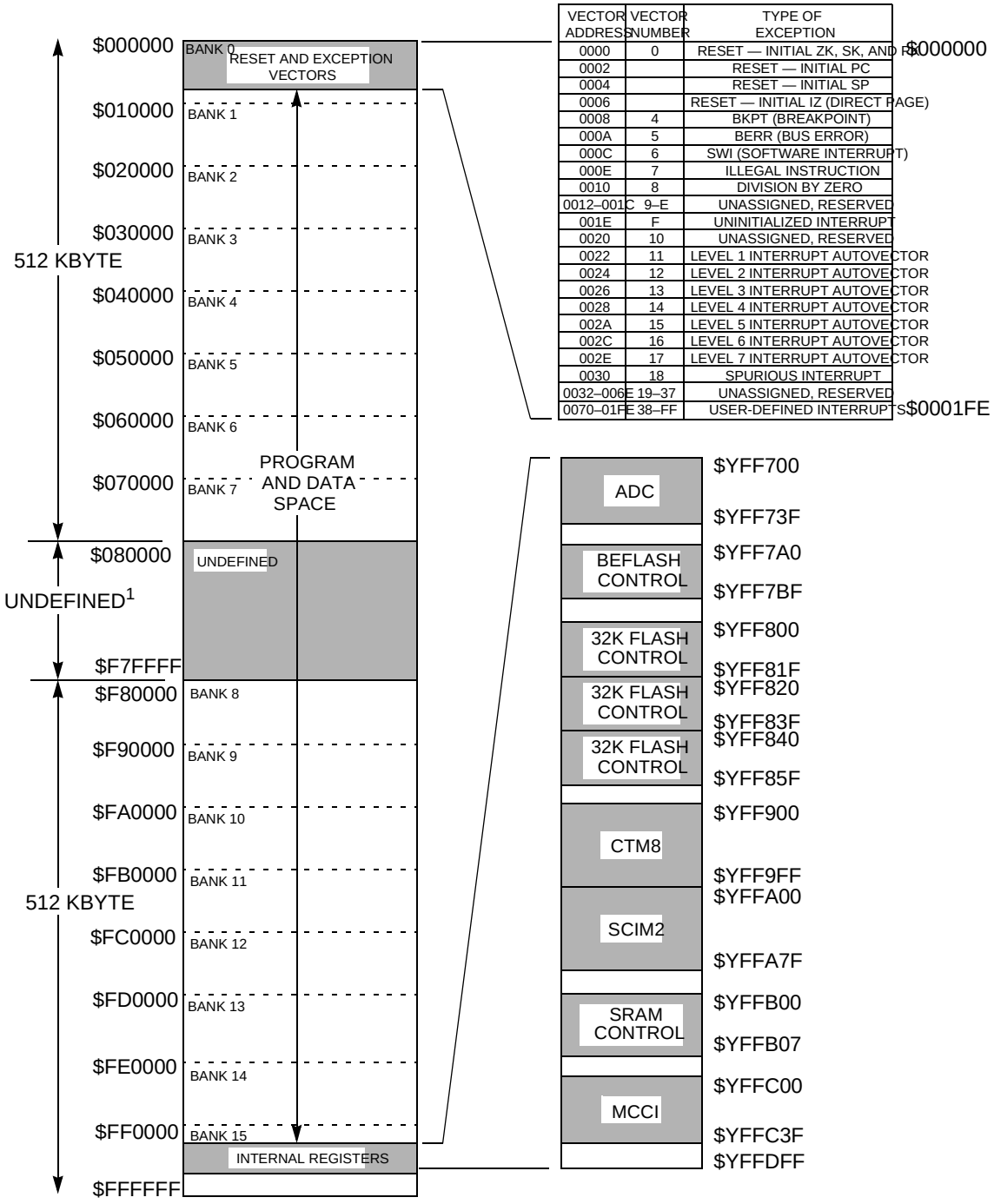
Figure 3-18 MC68HC16R3 Combined Program and Data Space Map



NOTE:

1. THE ADDRESSES DISPLAYED IN THIS MEMORY MAP ARE THE FULL 24- BIT IMB ADDRESSES. THE CPU16 ADDRESS BUS IS 20 BITS WIDE, AND CPU16 ADDRESS LINE 19 DRIVES IMB ADDRESS LINES [23:20]. THE BLOCK OF ADDRESSES FROM $080000 TO $F7FFFF MARKED AS UNDEFINED WILL NEVER APPEAR ON THE IMB. MEMORY BANKS 0 TO 15 APPEAR FULLY CONTIGUOUS IN THE CPU16'S FLAT 20-BIT ADDRESS SPACE. THE CPU16 NEED ONLY GENERATE A 20-BIT EFFECTIVE ADDRESS TO ACCESS ANY LOCATION IN THIS RANGE.

Figure 3-19 MC68HC16R3 Separate Program and Data Space Map

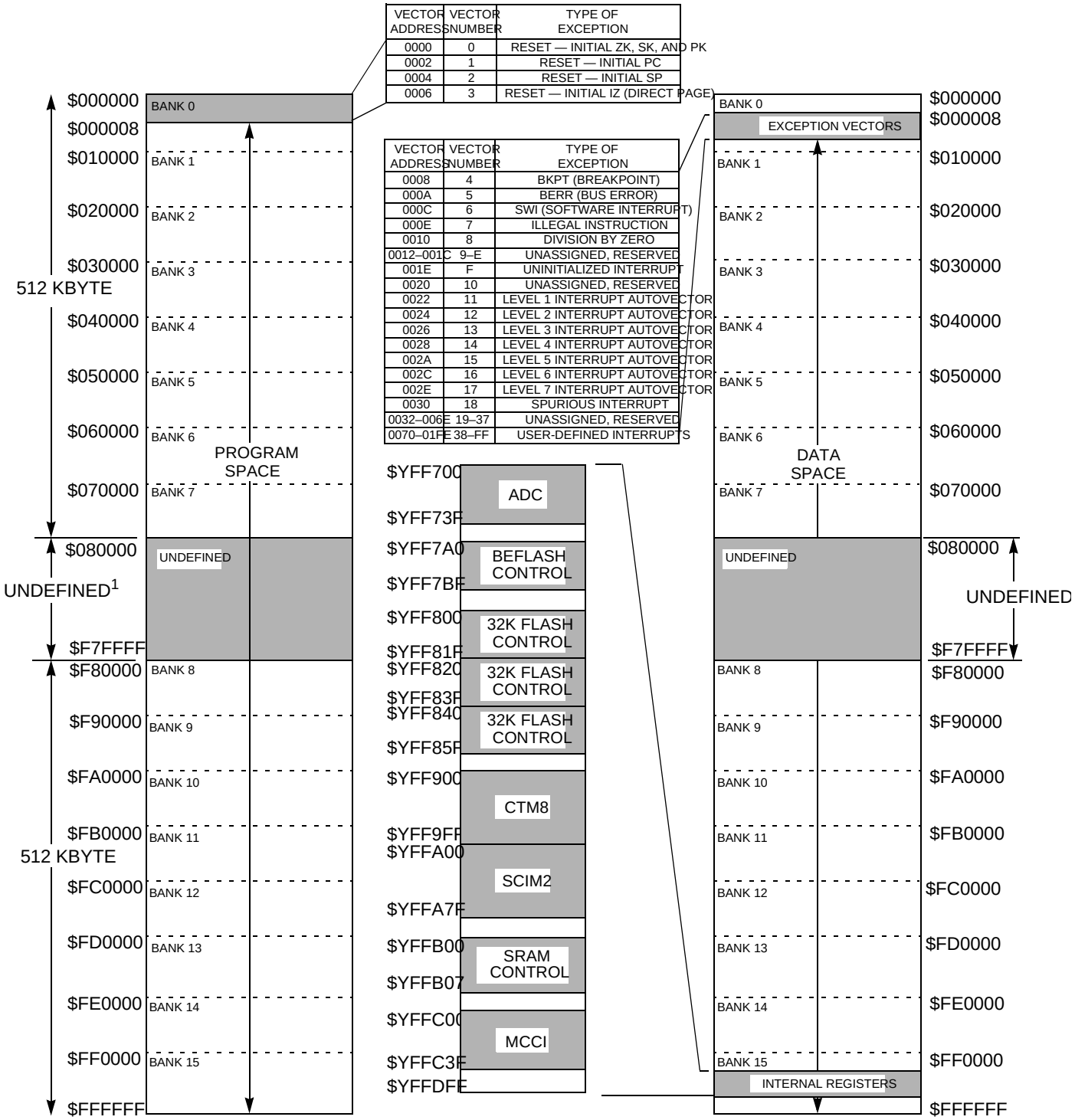


NOTE:

1. THE ADDRESSES DISPLAYED IN THIS MEMORY MAP ARE THE FULL 24- BIT IMB ADDRESSES. THE CPU16 ADDRESS BUS IS 20 BITS WIDE, AND CPU16 ADDRESS LINE 19 DRIVES IMB ADDRESS LINES [23:20]. THE BLOCK OF ADDRESSES FROM $080000 TO $F7FFFF MARKED AS UNDEFINED WILL NEVER APPEAR ON THE IMB. MEMORY BANKS 0 TO 15 APPEAR FULLY CONTIGUOUS IN THE CPU16'S FLAT 20-BIT ADDRESS SPACE. THE CPU16 NEED ONLY GENERATE A 20-BIT EFFECTIVE ADDRESS TO ACCESS ANY LOCATION IN THIS RANGE.

916R3 MEM MAP (C)

Figure 3-20 MC68HC916R3 Combined Program and Data Space Map



NOTE:
1. THE ADDRESSES DISPLAYED IN THIS MEMORY MAP ARE THE FULL 24- BIT IMB ADDRESSES. THE CPU16 ADDRESS BUS IS 20 BITS WIDE, AND CPU16 ADDRESS LINE 19 DRIVES IMB ADDRESS LINES [23:20]. THE BLOCK OF ADDRESSES FROM $080000 TO $F7FFFF MARKED AS UNDEFINED WILL NEVER APPEAR ON THE IMB. MEMORY BANKS 0 TO 15 APPEAR FULLY CONTIGUOUS IN THE CPU16'S FLAT 20-BIT ADDRESS SPACE. THE CPU16 NEED ONLY GENERATE A 20-BIT EFFECTIVE ADDRESS TO ACCESS ANY LOCATION IN THIS RANGE.

Figure 3-21 MC68HC916R3 Separate Program and Data Space Map

SECTION 4
CENTRAL PROCESSOR UNIT

This section is an overview of the central processor unit (CPU16). For detailed information, refer to the *CPU16 Reference Manual* (CPU16RM/AD).

4.1 General

The CPU16 provides compatibility with the M68HC11 CPU and also provides additional capabilities associated with 16- and 32-bit data sizes, 20-bit addressing, and digital signal processing. CPU16 registers are an integral part of the CPU and are not addressed as memory locations.

The CPU16 treats all peripheral, I/O, and memory locations as parts of a linear 1 Megabyte address space. There are no special instructions for I/O that are separate from instructions for addressing memory. Address space is made up of sixteen 64-Kbyte banks. Specialized bank addressing techniques and support registers provide transparent access across bank boundaries.

The CPU16 interacts with external devices and with other modules within the microcontroller via a standardized bus and bus interface. There are bus protocols used for memory and peripheral accesses, as well as for managing a hierarchy of interrupt priorities.

4.2 Register Model

Figure 4-1 shows the CPU16 register model. Refer to the paragraphs that follow for a detailed description of each register.

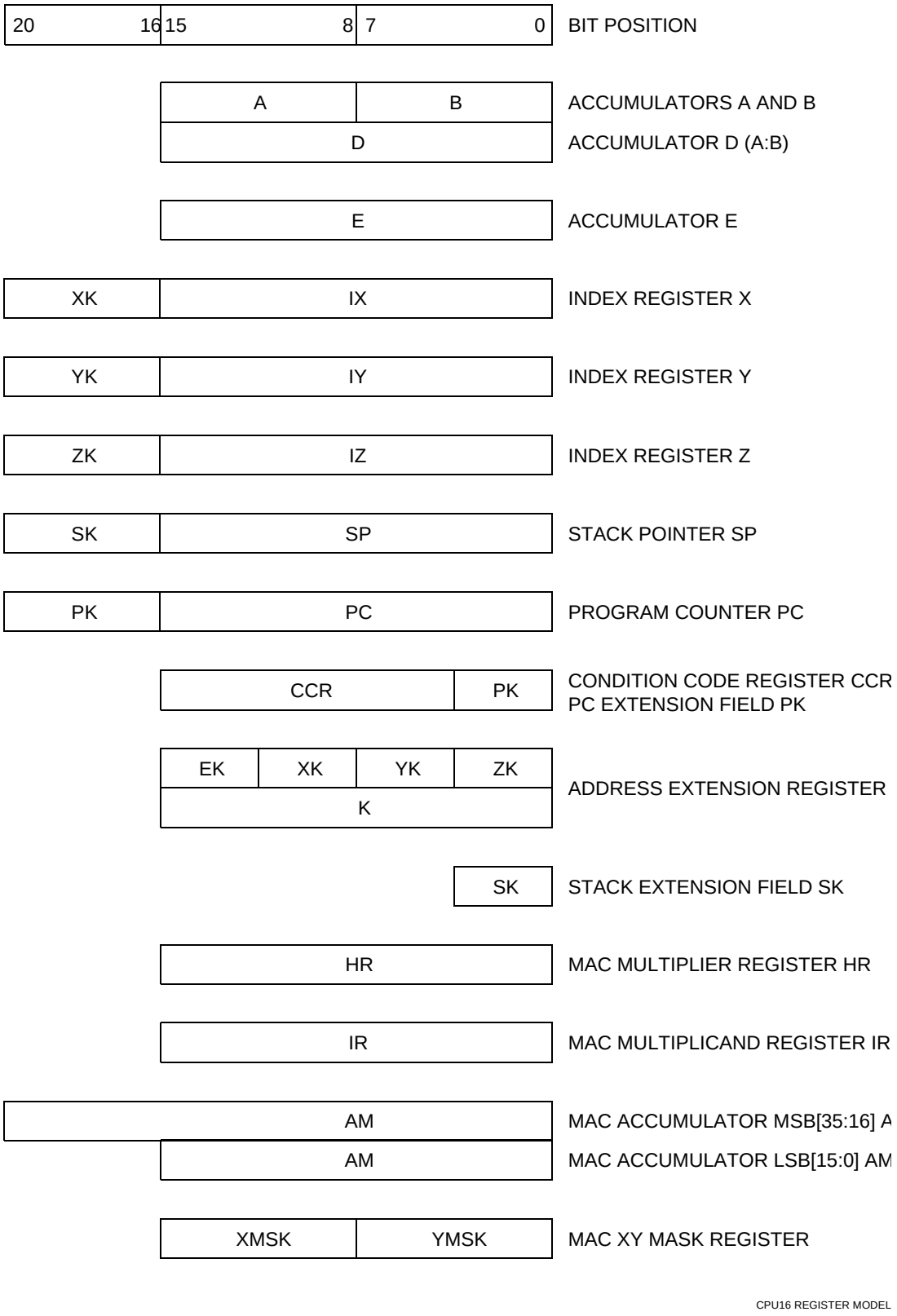


Figure 4-1 CPU16 Register Model

4.2.1 Accumulators

The CPU16 has two 8-bit accumulators (A and B) and one 16-bit accumulator (E). In addition, accumulators A and B can be concatenated into a second 16-bit double accumulator (D).

Accumulators A, B, and D are general-purpose registers that hold operands and results during mathematical and data manipulation operations.

Accumulator E, which can be used in the same way as accumulator D, also extends CPU16 capabilities. It allows more data to be held within the CPU16 during operations, simplifies 32-bit arithmetic and digital signal processing, and provides a practical 16-bit accumulator offset indexed addressing mode.

4.2.2 Index Registers

The CPU16 has three 16-bit index registers (IX, IY, and IZ). Each index register has an associated 4-bit extension field (XK, YK, and ZK).

Concatenated registers and extension fields provide 20-bit indexed addressing and support data structure functions anywhere in the CPU16 address space.

IX and IY can perform the same operations as M68HC11 registers of the same names, but the CPU16 instruction set provides additional indexed operations.

IZ can perform the same operations as IX and IY. IZ also provides an additional indexed addressing capability that replaces M68HC11 direct addressing mode. Initial IZ and ZK extension field values are included in the RESET exception vector, so that ZK:IZ can be used as a direct page pointer out of reset.

4.2.3 Stack Pointer

The CPU16 stack pointer (SP) is 16 bits wide. An associated 4-bit extension field (SK) provides 20-bit stack addressing.

Stack implementation in the CPU16 is from high to low memory. The stack grows downward as it is filled. SK:SP are decremented each time data is pushed on the stack, and incremented each time data is pulled from the stack.

SK:SP point to the next available stack address rather than to the address of the latest stack entry. Although the stack pointer is normally incremented or decremented by word address, it is possible to push and pull byte-sized data. Setting the stack pointer to an odd value causes data misalignment, which reduces performance.

4.2.4 Program Counter

The CPU16 program counter (PC) is 16 bits wide. An associated 4-bit extension field (PK) provides 20-bit program addressing.

CPU16 instructions are fetched from even word boundaries. Address line 0 always has a value of zero during instruction fetches to ensure that instructions are fetched from word-aligned addresses.

4.2.5 Condition Code Register

The 16-bit condition code register is composed of two functional blocks. The eight MSB, which correspond to the CCR on the M68HC11, contain the low-power stop control bit and processor status flags. The eight LSB contain the interrupt priority field, the DSP saturation mode control bit, and the program counter address extension field.

Figure 4-2 shows the condition code register. Detailed descriptions of each status indicator and field in the register follow the figure.

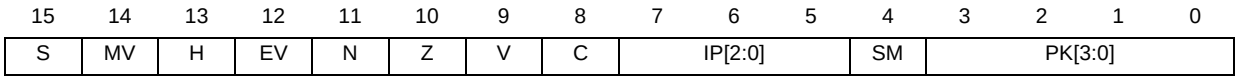


Figure 4-2 Condition Code Register

S — STOP Enable
0 = Stop clock when LPSTOP instruction is executed
1 = Perform NOP when LPSTOP instruction is executed

MV — Accumulator M overflow flag
MV is set when an overflow into AM35 has occurred.

H — Half Carry Flag
H is set when a carry from A3 or B3 occurs during BCD addition.

EV — Accumulator M Extension Overflow Flag
EV is set when an overflow into AM31 has occurred.

N — Negative Flag
N is set under the following conditions:
- When the MSB is set in the operand of a read operation.
- When the MSB is set in the result of a logic or arithmetic operation.

Z — Zero Flag
Z is set under the following conditions:
- When all bits are zero in the operand of a read operation.
- When all bits are zero in the result of a logic or arithmetic operation.

V — Overflow Flag
V is set when a two’s complement overflow occurs as the result of an operation.

C — Carry Flag
C is set when a carry or borrow occurs during an arithmetic operation. This flag is also used during shift and rotate to facilitate multiple word operations.

IP[2:0] — Interrupt Priority Field
The priority value in this field (0 to 7) is used to mask interrupts.

SM — Saturate Mode Bit

When SM is set and either EV or MV is set, data read from AM using TMER or TMET is given maximum positive or negative value, depending on the state of the AM sign bit before overflow.

PK[3:0] — Program Counter Address Extension Field

This field is concatenated with the program counter to form a 20-bit address.

4.2.6 Address Extension Register and Address Extension Fields

There are six 4-bit address extension fields. EK, XK, YK, and ZK are contained by the address extension register (K), PK is part of the CCR, and SK stands alone.

Extension fields are the bank portions of 20-bit concatenated bank:byte addresses used in the CPU16 linear memory management scheme.

All extension fields except EK correspond directly to a register. XK, YK, and ZK extend registers IX, IY, and IZ. PK extends the PC; and SK extends the SP. EK holds the four MSB of the 20-bit address used by the extended addressing mode.

4.2.7 Multiply and Accumulate Registers

The multiply and accumulate (MAC) registers are part of a CPU submodule that performs repetitive signed fractional multiplication and stores the cumulative result. These operations are part of control-oriented digital signal processing.

There are four MAC registers. Register H contains the 16-bit signed fractional multiplier. Register I contains the 16-bit signed fractional multiplicand. Accumulator M is a specialized 36-bit product accumulation register. XMSK and YMSK contain 8-bit mask values used in modulo addressing.

The CPU16 has a special subset of signal processing instructions that manipulate the MAC registers and perform signal processing calculations.

4.3 Memory Management

The CPU16 provides a 1-Mbyte address space. There are 16 banks within the address space. Each bank is made up of 64 Kbytes addressed from $0000 to $FFFF. Banks are selected by means of the address extension fields associated with individual CPU16 registers.

In addition, address space can be split into discrete 1-Mbyte program and data spaces by externally decoding the MCU’s function code outputs. When this technique is used, instruction fetches and reset vector fetches access program space, while exception vector fetches (other than for reset), data accesses, and stack accesses are made in data space.

4.3.1 Address Extension

All CPU16 resources used to generate addresses are effectively 20 bits wide. These resources include the index registers, program counter, and stack pointer. All addressing modes use 20-bit addresses.

Twenty-bit addresses are formed from a 16-bit byte address generated by an individual CPU16 register and a 4-bit address extension contained in an associated extension field. The byte address corresponds to ADDR[15:0] and the address extension corresponds to ADDR[19:16].

4.3.2 Extension Fields

Each of the six address extension fields is used for a different type of access. All but EK are associated with particular CPU16 registers. There are several ways to manipulate extension fields and the address map. Refer to the *CPU16 Reference Manual* (CPU16RM/AD) for detailed information.

4.4 Data Types

The CPU16 uses the following types of data:

- Bits
- 4-bit signed integers
- 8-bit (byte) signed and unsigned integers
- 8-bit, 2-digit binary coded decimal (BCD) numbers
- 16-bit (word) signed and unsigned integers
- 32-bit (long word) signed and unsigned integers
- 16-bit signed fractions
- 32-bit signed fractions
- 36-bit signed fixed-point numbers
- 20-bit effective addresses

There are 8 bits in a byte and 16 bits in a word. Bit set and clear instructions use both byte and word operands. Bit test instructions use byte operands.

Negative integers are represented in two’s complement form. 4-bit signed integers, packed two to a byte, are used only as X and Y offsets in MAC and RMAC operations. 32-bit integers are used only by extended multiply and divide instructions, and by the associated LDED and STED instructions.

BCD numbers are packed, two digits per byte. BCD operations use byte operands.

Signed 16-bit fractions are used by the fractional multiplication instructions, and as multiplicand and multiplier operands in the MAC unit. Bit 15 is the sign bit, and there is an implied radix point between bits 15 and 14. There are 15 bits of magnitude. The range of values is – 1 ($8000) to $1 - 2^{-15}$ ($7FFF).

Signed 32-bit fractions are used only by the fractional multiplication and division instructions. Bit 31 is the sign bit. An implied radix point lies between bits 31 and 30. There are 31 bits of magnitude. The range of values is – 1 ($80000000) to $1 - 2^{-31}$ ($7FFFFFFF).

Signed 36-bit fixed-point numbers are used only by the MAC unit. Bit 35 is the sign bit. Bits [34:31] are sign extension bits. There is an implied radix point between bits 31 and 30. There are 31 bits of magnitude, but use of the extension bits allows representation of numbers in the range – 16 ($800000000) to 15.999969482 ($7FFFFFFFF).

4.5 Memory Organization

Both program and data memory are divided into sixteen 64-Kbyte banks. Addressing is linear. A 20-bit extended address can access any byte location in the appropriate address space.

A word is composed of two consecutive bytes. A word address is normally an even byte address. Byte 0 of a word has a lower 16-bit address than byte 1. Long words and 32-bit signed fractions consist of two consecutive words, and are normally accessed at the address of byte 0 in word 0.

Instruction fetches always access word addresses. Word operands are normally accessed at even byte addresses, but can be accessed at odd byte addresses, with a substantial performance penalty.

To permit compatibility with the M68HC11, misaligned word transfers and misaligned stack accesses are allowed. Transferring a misaligned word requires two successive byte transfer operations.

Figure 4-3 shows how each CPU16 data type is organized in memory. Consecutive even addresses show size and alignment.

Address	Type															
$0000	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
$0002	BYTE0								BYTE1							
$0004	±	X OFFSET			±	Y OFFSET			±	X OFFSET			±	Y OFFSET		
$0006	BCD1				BCD0				BCD1				BCD0			
$0008	WORD 0															
$000A	WORD1															
$000C	MSW LONG WORD 0															
$000E	LSW LONG WORD 0															
$0010	MSW LONG WORD 1															
$0012	LSW LONG WORD 1															
$0014	±	⇐ (Radix Point)							16-BIT SIGNED FRACTION 0							
$0016	±	⇐ (Radix Point)							16-BIT SIGNED FRACTION 1							
$0018	±	⇐ (Radix Point)							MSW 32-BIT SIGNED FRACTION 0							
$001A	LSW 32-BIT SIGNED FRACTION 0															0
$001C	±	⇐ (Radix Point)							MSW 32-BIT SIGNED FRACTION 1							
$001E	LSW 32-BIT SIGNED FRACTION 1															0

MAC Data Types

35			32	31		16
±	«	«	«	«	⇐ (Radix Point)	MSW 32-BIT SIGNED FRACTION

15		0
LSW 32-BIT SIGNED FRACTION		
±	⇐ (Radix Point)	16-BIT SIGNED FRACTION

Address Data Type

19	16	15	0
4-Bit Address Extension		16-Bit Byte Address	

Figure 4-3 Data Types and Memory Organization

4.6 Addressing Modes

The CPU16 uses nine types of addressing. There are one or more addressing modes within each type. **Table 4-1** shows the addressing modes.

Table 4-1 Addressing Modes

Mode	Mnemonic	Description
Accumulator Offset	E,X	Index register X with accumulator E offset
	E,Y	Index register Y with accumulator E offset
	E,Z	Index register Z with accumulator E offset
Extended	EXT	Extended
	EXT20	20-bit extended
Immediate	IMM8	8-bit immediate
	IMM16	16-bit immediate
Indexed 8-Bit	IND8, X	Index register X with unsigned 8-bit offset
	IND8, Y	Index register Y with unsigned 8-bit offset
	IND8, Z	Index register Z with unsigned 8-bit offset
Indexed 16-Bit	IND16, X	Index register X with signed 16-bit offset
	IND16, Y	Index register Y with signed 16-bit offset
	IND16, Z	Index register Z with signed 16-bit offset
Indexed 20-Bit	IND20, X	Index register X with signed 20-bit offset
	IND20, Y	Index register Y with signed 20-bit offset
	IND20, Z	Index register Z with signed 20-bit offset
Inherent	INH	Inherent
Post-Modified Index	IXP	Signed 8-bit offset added to index register X after effective address is used
Relative	REL8	8-bit relative
	REL16	16-bit relative

All modes generate ADDR[15:0]. This address is combined with ADDR[19:16] from an operand or an extension field to form a 20-bit effective address.

NOTE

Access across 64-Kbyte address boundaries is transparent. ADDR[19:16] of the effective address are changed to make an access across a bank boundary. Extension field values will not change as a result of effective address computation.

4.6.1 Immediate Addressing Modes

In the immediate modes, an argument is contained in a byte or word immediately following the instruction. For IMM8 and IMM16 modes, the effective address is the address of the argument.

There are three specialized forms of IMM8 addressing.

- The AIS, AIX, AIY, AIZ, ADDD, and ADDE instructions decrease execution time by sign-extending the 8-bit immediate operand to 16 bits, then adding it to an appropriate register.
- The MAC and RMAC instructions use an 8-bit immediate operand to specify two signed 4-bit index register offsets.

- The PSHM and PULM instructions use an 8-bit immediate mask operand to indicate which registers must be pushed to or pulled from the stack.

4.6.2 Extended Addressing Modes

Regular extended mode instructions contain ADDR[15:0] in the word following the opcode. The effective address is formed by concatenating the EK field and the 16-bit byte address. EXT20 mode is used only by the JMP and JSR instructions. These instructions contain a 20-bit effective address that is zero-extended to 24 bits to give the instruction an even number of bytes.

4.6.3 Indexed Addressing Modes

In the indexed modes, registers IX, IY, and IZ, together with their associated extension fields, are used to calculate the effective address.

For 8-bit indexed modes an 8-bit unsigned offset contained in the instruction is added to the value contained in an index register and its extension field.

For 16-bit modes, a 16-bit signed offset contained in the instruction is added to the value contained in an index register and its extension field.

For 20-bit modes, a 20-bit signed offset (zero-extended to 24 bits) is added to the value contained in an index register. These modes are used for JMP and JSR instructions only.

4.6.4 Inherent Addressing Mode

Inherent mode instructions use information directly available to the processor to determine the effective address. Operands, if any, are system resources and are thus not fetched from memory.

4.6.5 Accumulator Offset Addressing Mode

Accumulator offset modes form an effective address by sign-extending the content of accumulator E to 20 bits, then adding the result to an index register and its associated extension field. This mode allows use of an index register and an accumulator within a loop without corrupting accumulator D.

4.6.6 Relative Addressing Modes

Relative modes are used for branch and long branch instructions. If a branch condition is satisfied, a byte or word signed two’s complement offset is added to the concatenated PK field and program counter. The new PK:PC value is the effective address.

4.6.7 Post-Modified Index Addressing Mode

Post-modified index mode is used by the MOVB and MOVW instructions. A signed 8-bit offset is added to index register X after the effective address formed by XK:IX is used.

4.6.8 Use of CPU16 Indexed Mode to Replace M68HC11 Direct Mode

In M68HC11 systems, the direct addressing mode can be used to perform rapid accesses to RAM or I/O mapped from $0000 to $00FF. The CPU16 uses the first 512 bytes of Bank 0 for exception vectors. To provide an enhanced replacement for the MC68HC11's direct addressing mode, the ZK field and index register Z have been assigned reset initialization vectors. By resetting the ZK field to a chosen page and using indexed mode addressing, a programmer can access useful data structures anywhere in the address map.

4.7 Instruction Set

The CPU16 instruction set is based on the M68HC11 instruction set, but the opcode map has been rearranged to maximize performance with a 16-bit data bus. Most M68HC11 code can run on the CPU16 following reassembly. The user must take into account changed instruction times, the interrupt mask, and the changed interrupt stack frame (Refer to *Motorola Programming Note* M68HC16PN01/D, *Transporting M68HC11 Code to M68HC16 Devices*, for more information).

4.7.1 Instruction Set Summary

Table 4-2 is a quick reference to the entire CPU16 instruction set. Refer to the *CPU16 Reference Manual* (CPU16RM/AD) for detailed information about each instruction, assembler syntax, and condition code evaluation. **Table 4-3** provides a key to the table nomenclature.

Table 4-2 Instruction Set Summary

Mnemonic	Operation	Description	Address	Instruction			Condition Codes							
			Mode	Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
ABA	Add B to A	(A) + (B) ⇒ A	INH	370B	—	2	—	—	Δ	—	Δ	Δ	Δ	Δ
ABX	Add B to IX	(XK : IX) + (000 : B) ⇒ XK : IX	INH	374F	—	2	—	—	—	—	—	—	—	—
ABY	Add B to IY	(YK : IY) + (000 : B) ⇒ YK : IY	INH	375F	—	2	—	—	—	—	—	—	—	—
ABZ	Add B to IZ	(ZK : IZ) + (000 : B) ⇒ ZK : IZ	INH	376F	—	2	—	—	—	—	—	—	—	—
ACE	Add E to AM	(AM[31:16]) + (E) ⇒ AM	INH	3722	—	2	—	Δ	—	Δ	—	—	—	—
ACED	Add E : D to AM	(AM) + (E : D) ⇒ AM	INH	3723	—	4	—	Δ	—	Δ	—	—	—	—
ADCA	Add with Carry to A	(A) + (M) + C ⇒ A	IND8, X	43	ff	6	—	—	Δ	—	Δ	Δ	Δ	Δ
			IND8, Y	53	ff	6								
			IND8, Z	63	ff	6								
			IMM8	73	ii	2								
			IND16, X	1743	gggg	6								
			IND16, Y	1753	gggg	6								
			IND16, Z	1763	gggg	6								
			EXT	1773	hh ll	6								
			E, X	2743	—	6								
			E, Y	2753	—	6								
			E, Z	2763	—	6								
ADCB	Add with Carry to B	(B) + (M) + C ⇒ B	IND8, X	C3	ff	6	—	—	Δ	—	Δ	Δ	Δ	Δ
			IND8, Y	D3	ff	6								
			IND8, Z	E3	ff	6								
			IMM8	F3	ii	2								
			IND16, X	17C3	gggg	6								
			IND16, Y	17D3	gggg	6								
			IND16, Z	17E3	gggg	6								
			EXT	17F3	hh ll	6								
			E, X	27C3	—	6								
			E, Y	27D3	—	6								
			E, Z	27E3	—	6								
ADCD	Add with Carry to D	(D) + (M : M + 1) + C ⇒ D	IND8, X	83	ff	6	—	—	—	—	Δ	Δ	Δ	Δ
			IND8, Y	93	ff	6								
			IND8, Z	A3	ff	6								
			IMM16	37B3	jj kk	4								
			IND16, X	37C3	gggg	6								
			IND16, Y	37D3	gggg	6								
			IND16, Z	37E3	gggg	6								
			EXT	37F3	hh ll	6								
			E, X	2783	—	6								
			E, Y	2793	—	6								
			E, Z	27A3	—	6								
ADCE	Add with Carry to E	(E) + (M : M + 1) + C ⇒ E	IMM16	3733	jj kk	4	—	—	—	—	Δ	Δ	Δ	Δ
			IND16, X	3743	gggg	6								
			IND16, Y	3753	gggg	6								
			IND16, Z	3763	gggg	6								
			EXT	3773	hh ll	6								
ADDA	Add to A	(A) + (M) ⇒ A	IND8, X	41	ff	6	—	—	Δ	—	Δ	Δ	Δ	Δ
			IND8, Y	51	ff	6								
			IND8, Z	61	ff	6								
			IMM8	71	ii	2								
			IND16, X	1741	gggg	6								
			IND16, Y	1751	gggg	6								
			IND16, Z	1761	gggg	6								
			EXT	1771	hh ll	6								
			E, X	2741	—	6								
			E, Y	2751	—	6								
			E, Z	2761	—	6								

Table 4-2 Instruction Set Summary (Continued)

Mnemonic	Operation	Description	Address	Instruction			Condition Codes							
			Mode	Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
ADDB	Add to B	(B) + (M) ⇒ B	IND8, X	C1	ff	6	—	—	Δ	—	Δ	Δ	Δ	Δ
			IND8, Y	D1	ff	6								
			IND8, Z	E1	ff	6								
			IMM8	F1	ii	2								
			IND16, X	17C1	gggg	6								
			IND16, Y	17D1	gggg	6								
			IND16, Z	17E1	gggg	6								
			EXT	17F1	hh ll	6								
			E, X	27C1	—	6								
			E, Y	27D1	—	6								
			E, Z	27E1	—	6								
ADDD	Add to D	(D) + (M : M + 1) ⇒ D	IND8, X	81	ff	6	—	—	—	—	Δ	Δ	Δ	Δ
			IND8, Y	91	ff	6								
			IND8, Z	A1	ff	6								
			IMM8	FC	ii	2								
			IMM16	37B1	jj kk	4								
			IND16, X	37C1	gggg	6								
			IND16, Y	37D1	gggg	6								
			IND16, Z	37E1	gggg	6								
			EXT	37F1	hh ll	6								
			E, X	2781	—	6								
			E, Y	2791	—	6								
			E, Z	27A1	—	6								
ADDE	Add to E	(E) + (M : M + 1) ⇒ E	IMM8	7C	ii	2	—	—	—	—	Δ	Δ	Δ	Δ
			IMM16	3731	jj kk	4								
			IND16, X	3741	gggg	6								
			IND16, Y	3751	gggg	6								
			IND16, Z	3761	gggg	6								
			EXT	3771	hh ll	6								
ADE	Add D to E	(E) + (D) ⇒ E	INH	2778	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ADX	Add D to IX	(XK : IX) + (20 « D) ⇒ XK : IX	INH	37CD	—	2	—	—	—	—	—	—	—	—
ADY	Add D to IY	(YK : IY) + (20 « D) ⇒ YK : IY	INH	37DD	—	2	—	—	—	—	—	—	—	—
ADZ	Add D to IZ	(ZK : IZ) + (20 « D) ⇒ ZK : IZ	INH	37ED	—	2	—	—	—	—	—	—	—	—
AEX	Add E to IX	(XK : IX) + (20 « E) ⇒ XK : IX	INH	374D	—	2	—	—	—	—	—	—	—	—
AEY	Add E to IY	(YK : IY) + (20 « E) ⇒ YK : IY	INH	375D	—	2	—	—	—	—	—	—	—	—
AEZ	Add E to IZ	(ZK : IZ) + (20 « E) ⇒ ZK : IZ	INH	376D	—	2	—	—	—	—	—	—	—	—
AIS	Add Immediate Data to Stack Pointer	(SK : SP) + (20 « IMM) ⇒ SK : SP	IMM8	3F	ii	2	—	—	—	—	—	—	—	—
			IMM16	373F	jj kk	4								
AIX	Add Immediate Value to IX	(XK : IX) + (20 « IMM) ⇒ XK : IX	IMM8	3C	ii	2	—	—	—	—	—	Δ	—	—
			IMM16	373C	jj kk	4								
AIY	Add Immediate Value to IY	(YK : IY) + (20 « IMM) ⇒ YK : IY	IMM8	3D	ii	2	—	—	—	—	—	Δ	—	—
			IMM16	373D	jj kk	4								
AIZ	Add Immediate Value to IZ	(ZK : IZ) + (20 « IMM) ⇒ ZK : IZ	IMM8	3E	ii	2	—	—	—	—	—	Δ	—	—
			IMM16	373E	jj kk	4								
ANDA	AND A	(A) • (M) ⇒ A	IND8, X	46	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	56	ff	6								
			IND8, Z	66	ff	6								
			IMM8	76	ii	2								
			IND16, X	1746	gggg	6								
			IND16, Y	1756	gggg	6								
			IND16, Z	1766	gggg	6								
			EXT	1776	hh ll	6								
			E, X	2746	—	6								
			E, Y	2756	—	6								
			E, Z	2766	—	6								

Table 4-2 Instruction Set Summary (Continued)

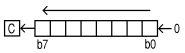
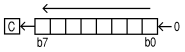
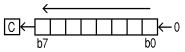
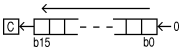
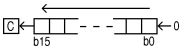
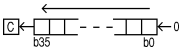
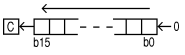
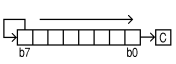
Mnemonic	Operation	Description	Address	Instruction			Condition Codes							
			Mode	Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
ANDB	AND B	(B) • (M) ⇒ B	IND8, X	C6	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	D6	ff	6								
			IND8, Z	E6	ff	6								
			IMM8	F6	ii	2								
			IND16, X	17C6	gggg	6								
			IND16, Y	17D6	gggg	6								
			IND16, Z	17E6	gggg	6								
			EXT	17F6	hh ll	6								
			E, X	27C6	—	6								
			E, Y	27D6	—	6								
			E, Z	27E6	—	6								
ANDD	AND D	(D) • (M : M + 1) ⇒ D	IND8, X	86	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	96	ff	6								
			IND8, Z	A6	ff	6								
			IMM16	37B6	jj kk	4								
			IND16, X	37C6	gggg	6								
			IND16, Y	37D6	gggg	6								
			IND16, Z	37E6	gggg	6								
			EXT	37F6	hh ll	6								
			E, X	2786	—	6								
			E, Y	2796	—	6								
			E, Z	27A6	—	6								
ANDE	AND E	(E) • (M : M + 1) ⇒ E	IMM16	3736	jj kk	4	—	—	—	—	Δ	Δ	0	—
			IND16, X	3746	gggg	6								
			IND16, Y	3756	gggg	6								
			IND16, Z	3766	gggg	6								
			EXT	3776	hh ll	6								
ANDP[1]	AND CCR	(CCR) • IMM16⇒ CCR	IMM16	373A	jj kk	4	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ
ASL	Arithmetic Shift Left		IND8, X	04	ff	8	—	—	—	—	Δ	Δ	Δ	Δ
			IND8, Y	14	ff	8								
			IND8, Z	24	ff	8								
			IND16, X	1704	gggg	8								
			IND16, Y	1714	gggg	8								
			IND16, Z	1724	gggg	8								
			EXT	1734	hh ll	8								
ASLA	Arithmetic Shift Left A		INH	3704	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ASLB	Arithmetic Shift Left B		INH	3714	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ASLD	Arithmetic Shift Left D		INH	27F4	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ASLE	Arithmetic Shift Left E		INH	2774	—	2	—	—	—	—	ý	ý	ý	ý
ASLM	Arithmetic Shift Left AM		INH	27B6	—	4	—	ý	—	ý	ý	—	—	ý
ASLW	Arithmetic Shift Left Word		IND16, X	2704	gggg	8	—	—	—	—	ý	ý	ý	ý
			IND16, Y	2714	gggg	8								
			IND16, Z	2724	gggg	8								
			EXT	2734	hh ll	8								
ASR	Arithmetic Shift Right		IND8, X	0D	ff	8	—	—	—	—	ý	ý	ý	ý
			IND8, Y	1D	ff	8								
			IND8, Z	2D	ff	8								
			IND16, X	170D	gggg	8								
			IND16, Y	171D	gggg	8								
			IND16, Z	172D	gggg	8								
			EXT	173D	hh ll	8								

Table 4-2 Instruction Set Summary (Continued)

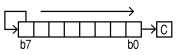
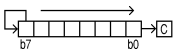
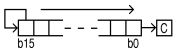
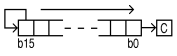
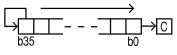
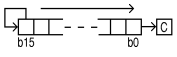
Mnemonic	Operation	Description	Address Mode	Instruction Opcode	Instruction Operand	Instruction Cycles	Condition Codes S	MV	H	EV	N	Z	V	C
ASRA	Arithmetic Shift Right A		INH	370D	—	2	—	—	—	—	ý	ý	ý	ý
ASRB	Arithmetic Shift Right B		INH	371D	—	2	—	—	—	—	ý	ý	ý	ý
ASRD	Arithmetic Shift Right D		INH	27FD	—	2	—	—	—	—	ý	ý	ý	ý
ASRE	Arithmetic Shift Right E		INH	277D	—	2	—	—	—	—	ý	ý	ý	ý
ASRM	Arithmetic Shift Right AM		INH	27BA	—	4	—	—	—	ý	ý	—	—	ý
ASRW	Arithmetic Shift Right Word		IND16, X IND16, Y IND16, Z EXT	270D 271D 272D 273D	gggg gggg gggg hh ll	8 8 8 8	—	—	—	—	ý	ý	ý	ý
BCC[2]	Branch if Carry Clear	If C = 0, branch	REL8	B4	rr	6, 2	—	—	—	—	—	—	—	—
BCLR	Clear Bit(s)	(M) • ($\overline{\text{Mask}}$) ⇒ M	IND8, X IND8, Y IND8, Z IND16, X IND16, Y IND16, Z EXT	1708 1718 1728 08 18 28 38	mm ff mm ff mm ff mm gggg mm gggg mm gggg mm hh ll	8 8 8 8 8 8 8	—	—	—	—	Δ	Δ	0	—
BCLRW	Clear Bit(s) in a Word	(M : M + 1) • ($\overline{\text{Mask}}$) ⇒ M : M + 1	IND16, X IND16, Y IND16, Z EXT	2708 2718 2728 2738	gggg mmmm gggg mmmm gggg mmmm hh ll mmmm	10 10 10 10	—	—	—	—	Δ	Δ	0	—
BCS	Branch if Carry Set	If C = 1, branch	REL8	B5	rr	6, 2	—	—	—	—	—	—	—	—
BEQ	Branch if Equal	If Z = 1, branch	REL8	B7	rr	6, 2	—	—	—	—	—	—	—	—
BGE	Branch if Greater Than or Equal to Zero	If N ⊕ V = 0, branch	REL8	BC	rr	6, 2	—	—	—	—	—	—	—	—
BGND	Enter Background Debug Mode	If BDM enabled, begin debug; else, illegal instruction trap	INH	37A6	—	—	—	—	—	—	—	—	—	—
BGT	Branch if Greater Than Zero	If Z ✣ (N ⊕ V) = 0, branch	REL8	BE	rr	6, 2	—	—	—	—	—	—	—	—
BHI	Branch if Higher	If C ✣ Z = 0, branch	REL8	B2	rr	6, 2	—	—	—	—	—	—	—	—
BITA	Bit Test A	(A) • (M)	IND8, X IND8, Y IND8, Z IMM8 IND16, X IND16, Y IND16, Z EXT E, X E, Y E, Z	49 59 69 79 1749 1759 1769 1779 2749 2759 2769	ff ff ff ii gggg gggg gggg hh ll — — —	6 6 6 2 6 6 6 6 6 6 6	—	—	—	—	Δ	Δ	0	—

Table 4-2 Instruction Set Summary (Continued)

Mnemonic	Operation	Description	Address	Instruction			Condition Codes							
			Mode	Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
BITB	Bit Test B	(B) • (M)	IND8, X IND8, Y IND8, Z IMM8 IND16, X IND16, Y IND16, Z EXT E, X E, Y E, Z	C9 D9 E9 F9 17C9 17D9 17E9 17F9 27C9 27D9 27E9	ff ff ff ii gggg gggg gggg hh ll — — —	6 6 6 2 6 6 6 6 6 6 6	—	—	—	—	Δ	Δ	0	—
BLE	Branch if Less Than or Equal to Zero	If Z ✣ (N ⊕ V) = 1, branch	REL8	BF	rr	6, 2	—	—	—	—	—	—	—	—
BLS	Branch if Lower or Same	If C ✣ Z = 1, branch	REL8	B3	rr	6, 2	—	—	—	—	—	—	—	—
BLT	Branch if Less Than Zero	If N ⊕ V = 1, branch	REL8	BD	rr	6, 2	—	—	—	—	—	—	—	—
BMI	Branch if Minus	If N = 1, branch	REL8	BB	rr	6, 2	—	—	—	—	—	—	—	—
BNE	Branch if Not Equal	If Z = 0, branch	REL8	B6	rr	6, 2	—	—	—	—	—	—	—	—
BPL	Branch if Plus	If N = 0, branch	REL8	BA	rr	6, 2	—	—	—	—	—	—	—	—
BRA	Branch Always	If 1 = 1, branch	REL8	B0	rr	6	—	—	—	—	—	—	—	—
BRCLR	Branch if Bit(s) Clear	If (M) • (Mask) = 0, branch	IND8, X IND8, Y IND8, Z IND16, X IND16, Y IND16, Z EXT	CB DB EB 0A 1A 2A 3A	mm ff rr mm ff rr mm ff rr mm gggg rrrr mm gggg rrrr mm gggg rrrr mm hh ll rrrr	10, 12 10, 12 10, 12 10, 14 10, 14 10, 14 10, 14	—	—	—	—	—	—	—	—
BRN	Branch Never	If 1 = 0, branch	REL8	B1	rr	2	—	—	—	—	—	—	—	—
BRSET	Branch if Bit(s) Set	If ($\overline{M}$) • (Mask) = 0, branch	IND8, X IND8, Y IND8, Z IND16, X IND16, Y IND16, Z EXT	8B 9B AB 0B 1B 2B 3B	mm ff rr mm ff rr mm ff rr mm gggg rrrr mm gggg rrrr mm gggg rrrr mm hh ll rrrr	10, 12 10, 12 10, 12 10, 14 10, 14 10, 14 10, 14	—	—	—	—	—	—	—	—
BSET	Set Bit(s)	(M) ✣ (Mask) ⇒ M	IND8, X IND8, Y IND8, Z IND16, X IND16, Y IND16, Z EXT	1709 1719 1729 09 19 29 39	mm ff mm ff mm ff mm gggg mm gggg mm gggg mm hh ll	8 8 8 8 8 8 8	—	—	—	—	Δ	Δ	0	Δ
BSETW	Set Bit(s) in Word	(M : M + 1) ✣ (Mask) ⇒ M : M + 1	IND16, X IND16, Y IND16, Z EXT	2709 2719 2729 2739	gggg mmmm gggg mmmm gggg mmmm hh ll mmmm	10 10 10 10	—	—	—	—	Δ	Δ	0	Δ
BSR	Branch to Subroutine	(PK : PC) - 2 ⇒ PK : PC Push (PC) (SK : SP) - 2 ⇒ SK : SP Push (CCR) (SK : SP) - 2 ⇒ SK : SP (PK : PC) + Offset ⇒ PK : PC	REL8	36	rr	10	—	—	—	—	—	—	—	—

Table 4-2 Instruction Set Summary (Continued)

Mnemonic	Operation	Description	Address	Instruction			Condition Codes							
			Mode	Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
BVC	Branch if Overflow Clear	If V = 0, branch	REL8	B8	rr	6, 2	—	—	—	—	—	—	—	—
BVS	Branch if Overflow Set	If V = 1, branch	REL8	B9	rr	6, 2	—	—	—	—	—	—	—	—
CBA	Compare A to B	(A) – (B)	INH	371B	—	2	—	—	—	—	Δ	Δ	Δ	Δ
CLR	Clear a Byte in Memory	$00 ⇒ M	IND8, X IND8, Y IND8, Z IND16, X IND16, Y IND16, Z EXT	05 15 25 1705 1715 1725 1735	ff ff ff gggg gggg gggg hh ll	4 4 4 6 6 6 6	—	—	—	—	0	1	0	0
CLRA	Clear A	$00 ⇒ A	INH	3705	—	2	—	—	—	—	0	1	0	0
CLRB	Clear B	$00 ⇒ B	INH	3715	—	2	—	—	—	—	0	1	0	0
CLRD	Clear D	$0000 ⇒ D	INH	27F5	—	2	—	—	—	—	0	1	0	0
CLRE	Clear E	$0000 ⇒ E	INH	2775	—	2	—	—	—	—	0	1	0	0
CLRM	Clear AM	$000000000 ⇒ AM[35:0]	INH	27B7	—	2	—	0	—	0	—	—	—	—
CLRW	Clear a Word in Memory	$0000 ⇒ M : M + 1	IND16, X IND16, Y IND16, Z EXT	2705 2715 2725 2735	gggg gggg gggg hh ll	6 6 6 6	—	—	—	—	0	1	0	0
CMPA	Compare A to Memory	(A) – (M)	IND8, X IND8, Y IND8, Z IMM8 IND16, X IND16, Y IND16, Z EXT E, X E, Y E, Z	48 58 68 78 1748 1758 1768 1778 2748 2758 2768	ff ff ff ii gggg gggg gggg hh ll — — —	6 6 6 2 6 6 6 6 6 6 6	—	—	—	—	Δ	Δ	Δ	Δ
CMPB	Compare B to Memory	(B) – (M)	IND8, X IND8, Y IND8, Z IMM8 IND16, X IND16, Y IND16, Z EXT E, X E, Y E, Z	C8 D8 E8 F8 17C8 17D8 17E8 17F8 27C8 27D8 27E8	ff ff ff ii gggg gggg gggg hh ll — — —	6 6 6 2 6 6 6 6 6 6 6	—	—	—	—	Δ	Δ	Δ	Δ
COM	One's Complement	$FF – (M) ⇒ M, or $\overline{M} ⇒ M$	IND8, X IND8, Y IND8, Z IND16, X IND16, Y IND16, Z EXT	00 10 20 1700 1710 1720 1730	ff ff ff gggg gggg gggg hh ll	8 8 8 8 8 8 8	—	—	—	—	Δ	Δ	0	1
COMA	One's Complement A	$FF – (A) ⇒ A, or $\overline{M} ⇒ A$	INH	3700	—	2	—	—	—	—	Δ	Δ	0	1
COMB	One's Complement B	$FF – (B) ⇒ B, or $\overline{B} ⇒ B$	INH	3710	—	2	—	—	—	—	Δ	Δ	0	1
COMD	One's Complement D	$FFFF – (D) ⇒ D, or $\overline{D} ⇒ D$	INH	27F0	—	2	—	—	—	—	Δ	Δ	0	1
COME	One's Complement E	$FFFF – (E) ⇒ E, or $\overline{E} ⇒ E$	INH	2770	—	2	—	—	—	—	Δ	Δ	0	1
COMW	One's Complement Word	$FFFF – M : M + 1 ⇒ M : M + 1, or $\overline{(M : M + 1)} ⇒ M : M + 1$	IND16, X IND16, Y IND16, Z EXT	2700 2710 2720 2730	gggg gggg gggg hh ll	8 8 8 8	—	—	—	—	Δ	Δ	0	1

Table 4-2 Instruction Set Summary (Continued)

Mnemonic	Operation	Description	Address	Instruction			Condition Codes							
			Mode	Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
CPD	Compare D to Memory	(D) – (M : M + 1)	IND8, X	88	ff	6	—	—	—	—	Δ	Δ	Δ	Δ
			IND8, Y	98	ff	6								
			IND8, Z	A8	ff	6								
			IMM16	37B8	jj kk	4								
			IND16, X	37C8	gggg	6								
			IND16, Y	37D8	gggg	6								
			IND16, Z	37E8	gggg	6								
			EXT	37F8	hh ll	6								
			E, X	2788	—	6								
			E, Y	2798	—	6								
			E, Z	27A8	—	6								
CPE	Compare E to Memory	(E) – (M : M + 1)	IMM16	3738	jjkk	4	—	—	—	—	Δ	Δ	Δ	Δ
			IND16, X	3748	gggg	6								
			IND16, Y	3758	gggg	6								
			IND16, Z	3768	gggg	6								
			EXT	3778	hhll	6								
CPS	Compare Stack Pointer to Memory	(SP) – (M : M + 1)	IND8, X	4F	ff	6	—	—	—	—	Δ	Δ	Δ	Δ
			IND8, Y	5F	ff	6								
			IND8, Z	6F	ff	6								
			IMM16	377F	jj kk	4								
			IND16, X	174F	gggg	6								
			IND16, Y	175F	gggg	6								
			IND16, Z	176F	gggg	6								
			EXT	177F	hh ll	6								
CPX	Compare IX to Memory	(IX) – (M : M + 1)	IND8, X	4C	ff	6	—	—	—	—	Δ	Δ	Δ	Δ
			IND8, Y	5C	ff	6								
			IND8, Z	6C	ff	6								
			IMM16	377C	jj kk	4								
			IND16, X	174C	gggg	6								
			IND16, Y	175C	gggg	6								
			IND16, Z	176C	gggg	6								
			EXT	177C	hh ll	6								
CPY	Compare IY to Memory	(IY) – (M : M + 1)	IND8, X	4D	ff	6	—	—	—	—	Δ	Δ	Δ	Δ
			IND8, Y	5D	ff	6								
			IND8, Z	6D	ff	6								
			IMM16	377D	jj kk	4								
			IND16, X	174D	gggg	6								
			IND16, Y	175D	gggg	6								
			IND16, Z	176D	gggg	6								
			EXT	177D	hh ll	6								
CPZ	Compare IZ to Memory	(IZ) – (M : M + 1)	IND8, X	4E	ff	6	—	—	—	—	ý	ý	ý	ý
			IND8, Y	5E	ff	6								
			IND8, Z	6E	ff	6								
			IMM16	377E	jj kk	4								
			IND16, X	174E	gggg	6								
			IND16, Y	175E	gggg	6								
			IND16, Z	176E	gggg	6								
			EXT	177E	hh ll	6								
DAA	Decimal Adjust A	$(A)_{10}$	INH	3721	—	2	—	—	—	—	Δ	Δ	U	Δ
DEC	Decrement Memory	(M) – $01 ⇒ M	IND8, X	01	ff	8	—	—	—	—	Δ	Δ	Δ	—
			IND8, Y	11	ff	8								
			IND8, Z	21	ff	8								
			IND16, X	1701	gggg	8								
			IND16, Y	1711	gggg	8								
			IND16, Z	1721	gggg	8								
			EXT	1731	hh ll	8								
DECA	Decrement A	(A) – $01 ⇒ A	INH	3701	—	2	—	—	—	—	Δ	Δ	Δ	—
DECB	Decrement B	(B) – $01 ⇒ B	INH	3711	—	2	—	—	—	—	Δ	Δ	Δ	—
DECW	Decrement Memory Word	(M : M + 1) – $0001 ⇒ M : M + 1	IND16, X	2701	gggg	8	—	—	—	—	Δ	Δ	Δ	—
			IND16, Y	2711	gggg	8								
			IND16, Z	2721	gggg	8								
			EXT	2731	hh ll	8								
EDIV	Extended Unsigned Integer Divide	(E : D) / (IX) Quotient ⇒ IX Remainder ⇒ D	INH	3728	—	24	—	—	—	—	Δ	Δ	Δ	Δ

Table 4-2 Instruction Set Summary (Continued)

Mnemonic	Operation	Description	Address	Instruction			Condition Codes							
			Mode	Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
EDIVS	Extended Signed Integer Divide	(E : D) / (IX) Quotient ⇒ IX Remainder ⇒ D	INH	3729	—	38	—	—	—	—	Δ	Δ	Δ	Δ
EMUL	Extended Unsigned Multiply	(E) * (D) ⇒ E : D	INH	3725	—	10	—	—	—	—	Δ	Δ	—	Δ
EMULS	Extended Signed Multiply	(E) * (D) ⇒ E : D	INH	3726	—	8	—	—	—	—	Δ	Δ	—	Δ
EORA	Exclusive OR A	(A) ⊕ (M) ⇒ A	IND8, X	44	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	54	ff	6								
			IND8, Z	64	ff	6								
			IMM8	74	ii	2								
			IND16, X	1744	gggg	6								
			IND16, Y	1754	gggg	6								
			IND16, Z	1764	gggg	6								
			EXT	1774	hh ll	6								
			E, X	2744	—	6								
			E, Y	2754	—	6								
			E, Z	2764	—	6								
EORB	Exclusive OR B	(B) ⊕ (M) ⇒ B	IND8, X	C4	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	D4	ff	6								
			IND8, Z	E4	ff	6								
			IMM8	F4	ii	2								
			IND16, X	17C4	gggg	6								
			IND16, Y	17D4	gggg	6								
			IND16, Z	17E4	gggg	6								
			EXT	17F4	hh ll	6								
			E, X	27C4	—	6								
			E, Y	27D4	—	6								
			E, Z	27E4	—	6								
EORD	Exclusive OR D	(D) ⊕ (M : M + 1) ⇒ D	IND8, X	84	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	94	ff	6								
			IND8, Z	A4	ff	6								
			IMM16	37B4	jj kk	4								
			IND16, X	37C4	gggg	6								
			IND16, Y	37D4	gggg	6								
			IND16, Z	37E4	gggg	6								
			EXT	37F4	hh ll	6								
			E, X	2784	—	6								
			E, Y	2794	—	6								
			E, Z	27A4	—	6								
EORE	Exclusive OR E	(E) ⊕ (M : M + 1) ⇒ E	IMM16	3734	jj kk	4	—	—	—	—	Δ	Δ	0	—
			IND16, X	3744	gggg	6								
			IND16, Y	3754	gggg	6								
			IND16, Z	3764	gggg	6								
			EXT	3774	hh ll	6								
FDIV	Fractional Unsigned Divide	(D) / (IX) ⇒ IX Remainder ⇒ D	INH	372B	—	22	—	—	—	—	—	Δ	Δ	Δ
FMULS	Fractional Signed Multiply	(E) * (D) ⇒ E : D[31:1] 0 ⇒ D[0]	INH	3727	—	8	—	—	—	—	Δ	Δ	Δ	Δ
IDIV	Integer Divide	(D) / (IX) ⇒ IX Remainder ⇒ D	INH	372A	—	22	—	—	—	—	—	Δ	0	Δ
INC	Increment Memory	(M) + $01 ⇒ M	IND8, X	03	ff	8	—	—	—	—	Δ	Δ	Δ	—
			IND8, Y	13	ff	8								
			IND8, Z	23	ff	8								
			IND16, X	1703	gggg	8								
			IND16, Y	1713	gggg	8								
			IND16, Z	1723	gggg	8								
			EXT	1733	hh ll	8								
INCA	Increment A	(A) + $01 ⇒ A	INH	3703	—	2	—	—	—	—	Δ	Δ	Δ	—
INCB	Increment B	(B) + $01 ⇒ B	INH	3713	—	2	—	—	—	—	Δ	Δ	Δ	—
INCW	Increment Memory Word	(M : M + 1) + $0001 ⇒ M : M + 1	IND16, X	2703	gggg	8	—	—	—	—	Δ	Δ	Δ	—
			IND16, Y	2713	gggg	8								
			IND16, Z	2723	gggg	8								
			EXT	2733	hh ll	8								

Table 4-2 Instruction Set Summary (Continued)

Mnemonic	Operation	Description	Address	Instruction			Condition Codes							
			Mode	Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
JMP	Jump	⟨ea⟩ ⇒ PK : PC	EXT20 IND20, X IND20, Y IND20, Z	7A 4B 5B 6B	zb hh ll zg gggg zg gggg zg gggg	6 8 8 8	—	—	—	—	—	—	—	—
JSR	Jump to Subroutine	Push (PC) (SK : SP) – $0002 ⇒ SK : SP Push (CCR) (SK : SP) – $0002 ⇒ SK : SP ⟨ea⟩ ⇒ PK : PC	EXT20 IND20, X IND20, Y IND20, Z	FA 89 99 A9	zb hh ll zg gggg zg gggg zg gggg	10 12 12 12	—	—	—	—	—	—	—	—
LBCC	Long Branch if Carry Clear	If C = 0, branch	REL16	3784	rrrr	6, 4	—	—	—	—	—	—	—	—
LBCS	Long Branch if Carry Set	If C = 1, branch	REL16	3785	rrrr	6, 4	—	—	—	—	—	—	—	—
LBEQ	Long Branch if Equal to Zero	If Z = 1, branch	REL16	3787	rrrr	6, 4	—	—	—	—	—	—	—	—
LBEV	Long Branch if EV Set	If EV = 1, branch	REL16	3791	rrrr	6, 4	—	—	—	—	—	—	—	—
LBGE	Long Branch if Greater Than or Equal to Zero	If N ⊕ V = 0, branch	REL16	378C	rrrr	6, 4	—	—	—	—	—	—	—	—
LBGT	Long Branch if Greater Than Zero	If Z ✣ (N ⊕ V) = 0, branch	REL16	378E	rrrr	6, 4	—	—	—	—	—	—	—	—
LBHI	Long Branch if Higher	If C ✣ Z = 0, branch	REL16	3782	rrrr	6, 4	—	—	—	—	—	—	—	—
LBLE	Long Branch if Less Than or Equal to Zero	If Z ✣ (N ⊕ V) = 1, branch	REL16	378F	rrrr	6, 4	—	—	—	—	—	—	—	—
LBLS	Long Branch if Lower or Same	If C ✣ Z = 1, branch	REL16	3783	rrrr	6, 4	—	—	—	—	—	—	—	—
LBLT	Long Branch if Less Than Zero	If N ⊕ V = 1, branch	REL16	378D	rrrr	6, 4	—	—	—	—	—	—	—	—
LBMI	Long Branch if Minus	If N = 1, branch	REL16	378B	rrrr	6, 4	—	—	—	—	—	—	—	—
LBMV	Long Branch if MV Set	If MV = 1, branch	REL16	3790	rrrr	6, 4	—	—	—	—	—	—	—	—
LBNE	Long Branch if Not Equal to Zero	If Z = 0, branch	REL16	3786	rrrr	6, 4	—	—	—	—	—	—	—	—
LBPL	Long Branch if Plus	If N = 0, branch	REL16	378A	rrrr	6, 4	—	—	—	—	—	—	—	—
LBRA	Long Branch Always	If 1 = 1, branch	REL16	3780	rrrr	6	—	—	—	—	—	—	—	—
LBRN	Long Branch Never	If 1 = 0, branch	REL16	3781	rrrr	6	—	—	—	—	—	—	—	—
LBSR	Long Branch to Subroutine	Push (PC) (SK : SP) – 2 ⇒ SK : SP Push (CCR) (SK : SP) – 2 ⇒ SK : SP (PK : PC) + Offset ⇒ PK : PC	REL16	27F9	rrrr	10	—	—	—	—	—	—	—	—
LBVC	Long Branch if Overflow Clear	If V = 0, branch	REL16	3788	rrrr	6, 4	—	—	—	—	—	—	—	—
LBVS	Long Branch if Overflow Set	If V = 1, branch	REL16	3789	rrrr	6, 4	—	—	—	—	—	—	—	—
LDAA	Load A	(M) ⇒ A	IND8, X IND8, Y IND8, Z IMM8 IND16, X IND16, Y IND16, Z EXT E, X E, Y E, Z	45 55 65 75 1745 1755 1765 1775 2745 2755 2765	ff ff ff ii gggg gggg gggg hh ll — — —	6 6 6 2 6 6 6 6 6 6 6	—	—	—	—	Δ	Δ	0	—

Table 4-2 Instruction Set Summary (Continued)

Mnemonic	Operation	Description	Address	Instruction			Condition Codes							
			Mode	Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
LDAB	Load B	(M) ⇒ B	IND8, X	C5	ff	6	—	—	—	—	Δ	Δ	0	Δ
			IND8, Y	D5	ff	6								
			IND8, Z	E5	ff	6								
			IMM8	F5	ii	2								
			IND16, X	17C5	gggg	6								
			IND16, Y	17D5	gggg	6								
			IND16, Z	17E5	gggg	6								
			EXT	17F5	hh ll	6								
			E, X	27C5	—	6								
			E, Y	27D5	—	6								
			E, Z	27E5	—	6								
LDD	Load D	(M : M + 1) ⇒ D	IND8, X	85	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	95	ff	6								
			IND8, Z	A5	ff	6								
			IMM16	37B5	jj kk	4								
			IND16, X	37C5	gggg	6								
			IND16, Y	37D5	gggg	6								
			IND16, Z	37E5	gggg	6								
			EXT	37F5	hh ll	6								
			E, X	2785	—	6								
			E, Y	2795	—	6								
			E, Z	27A5	—	6								
LDE	Load E	(M : M + 1) ⇒ E	IMM16	3735	jj kk	4	—	—	—	—	Δ	Δ	0	—
			IND16, X	3745	gggg	6								
			IND16, Y	3755	gggg	6								
			IND16, Z	3765	gggg	6								
			EXT	3775	hh ll	6								
LDED	Load Concatenated E and D	(M : M + 1) ⇒ E (M + 2 : M + 3) ⇒ D	EXT	2771	hh ll	8	—	—	—	—	—	—	—	—
LDHI	Initialize H and I	$(M : M + 1)_X \Rightarrow H R$ $(M : M + 1)_Y \Rightarrow I R$	INH	27B0	—	8	—	—	—	—	—	—	—	—
LDS	Load SP	(M : M + 1) ⇒ SP	IND8, X	CF	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	DF	ff	6								
			IND8, Z	EF	ff	6								
			IND16, X	17CF	gggg	6								
			IND16, Y	17DF	gggg	6								
			IND16, Z	17EF	gggg	6								
			EXT	17FF	hh ll	6								
			IMM16	37BF	jj kk	4								
LDX	Load IX	(M : M + 1) ⇒ IX	IND8, X	CC	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	DC	ff	6								
			IND8, Z	EC	ff	6								
			IMM16	37BC	jj kk	4								
			IND16, X	17CC	gggg	6								
			IND16, Y	17DC	gggg	6								
			IND16, Z	17EC	gggg	6								
			EXT	17FC	hh ll	6								
LDY	Load IY	(M : M + 1) ⇒ IY	IND8, X	CD	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	DD	ff	6								
			IND8, Z	ED	ff	6								
			IMM16	37BD	jj kk	4								
			IND16, X	17CD	gggg	6								
			IND16, Y	17DD	gggg	6								
			IND16, Z	17ED	gggg	6								
			EXT	17FD	hh ll	6								
LDZ	Load IZ	(M : M + 1) ⇒ IZ	IND8, X	CE	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	DE	ff	6								
			IND8, Z	EE	ff	6								
			IMM16	37BE	jj kk	4								
			IND16, X	17CE	gggg	6								
			IND16, Y	17DE	gggg	6								
			IND16, Z	17EE	gggg	6								
			EXT	17FE	hh ll	6								

Table 4-2 Instruction Set Summary (Continued)

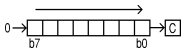
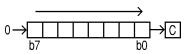
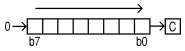
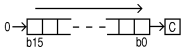
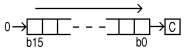
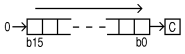
Mnemonic	Operation	Description	Address	Instruction			Condition Codes							
			Mode	Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
LPSTOP	Low Power Stop	If $\overline{S}$ then STOP else NOP	INH	27F1	—	4, 20	—	—	—	—	—	—	—	—
LSR	Logical Shift Right		IND8, X IND8, Y IND8, Z IND16, X IND16, Y IND16, Z EXT	0F 1F 2F 170F 171F 172F 173F	ff ff ff gggg gggg gggg hh ll	8 8 8 8 8 8 8	—	—	—	—	0	Δ	Δ	Δ
LSRA	Logical Shift Right A		INH	370F	—	2	—	—	—	—	0	Δ	Δ	Δ
LSRB	Logical Shift Right B		INH	371F	—	2	—	—	—	—	0	Δ	Δ	Δ
LSRD	Logical Shift Right D		INH	27FF	—	2	—	—	—	—	0	Δ	Δ	Δ
LSRE	Logical Shift Right E		INH	277F	—	2	—	—	—	—	0	Δ	Δ	Δ
LSRW	Logical Shift Right Word		IND16, X IND16, Y IND16, Z EXT	270F 271F 272F 273F	gggg gggg gggg hh ll	8 8 8 8	—	—	—	—	0	Δ	Δ	Δ
MAC	Multiply and Accumulate Signed 16-Bit Fractions	(HR) * (IR) ⇒ E : D (AM) + (E : D) ⇒ AM Qualified (IX) ⇒ IX Qualified (IY) ⇒ IY (HR) ⇒ IZ $(M : M + 1)_X \Rightarrow HR$ $(M : M + 1)_Y \Rightarrow IR$	IMM8	7B	xoyo	12	—	Δ	—	Δ	—	—	Δ	—
MOVB	Move Byte	$(M_1) \Rightarrow M_2$	IXP to EXT EXT to IXP EXT to EXT	30 32 37FE	ff hh ll ff hh ll hh ll hh ll	8 8 10	—	—	—	—	Δ	Δ	0	—
MOVW	Move Word	$(M : M + 1_1) \Rightarrow M : M + 1_2$	IXP to EXT EXT to IXP EXT to EXT	31 33 37FF	ff hh ll ff hh ll hh ll hh ll	8 8 10	—	—	—	—	Δ	Δ	0	—
MUL	Multiply	(A) * (B) ⇒ D	INH	3724	—	10	—	—	—	—	—	—	—	Δ
NEG	Negate Memory	$00 – (M) ⇒ M	IND8, X IND8, Y IND8, Z IND16, X IND16, Y IND16, Z EXT	02 12 22 1702 1712 1722 1732	ff ff ff gggg gggg gggg hh ll	8 8 8 8 8 8 8	—	—	—	—	Δ	Δ	Δ	Δ
NEGA	Negate A	$00 – (A) ⇒ A	INH	3702	—	2	—	—	—	—	Δ	Δ	Δ	Δ
NEGB	Negate B	$00 – (B) ⇒ B	INH	3712	—	2	—	—	—	—	Δ	Δ	Δ	Δ
NEGD	Negate D	$0000 – (D) ⇒ D	INH	27F2	—	2	—	—	—	—	Δ	Δ	Δ	Δ
NEGE	Negate E	$0000 – (E) ⇒ E	INH	2772	—	2	—	—	—	—	Δ	Δ	Δ	Δ
NEGW	Negate Memory Word	$0000 – (M : M + 1) ⇒ M : M + 1	IND16, X IND16, Y IND16, Z EXT	2702 2712 2722 2732	gggg gggg gggg hh ll	8 8 8 8	—	—	—	—	Δ	Δ	Δ	Δ
NOP	Null Operation	—	INH	274C	—	2	—	—	—	—	—	—	—	—

Table 4-2 Instruction Set Summary (Continued)

Mnemonic	Operation	Description	Address	Instruction			Condition Codes							
			Mode	Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
ORAA	OR A	(A) ✣ (M) ⇒ A	IND8, X	47	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	57	ff	6								
			IND8, Z	67	ff	6								
			IMM8	77	ii	2								
			IND16, X	1747	gggg	6								
			IND16, Y	1757	gggg	6								
			IND16, Z	1767	gggg	6								
			EXT	1777	hh ll	6								
			E, X	2747	—	6								
			E, Y	2757	—	6								
			E, Z	2767	—	6								
ORAB	OR B	(B) ✣ (M) ⇒ B	IND8, X	C7	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	D7	ff	6								
			IND8, Z	E7	ff	6								
			IMM8	F7	ii	2								
			IND16, X	17C7	gggg	6								
			IND16, Y	17D7	gggg	6								
			IND16, Z	17E7	gggg	6								
			EXT	17F7	hh ll	6								
			E, X	27C7	—	6								
			E, Y	27D7	—	6								
			E, Z	27E7	—	6								
ORD	OR D	(D) ✣ (M : M + 1) ⇒ D	IND8, X	87	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	97	ff	6								
			IND8, Z	A7	ff	6								
			IMM16	37B7	jj kk	4								
			IND16, X	37C7	gggg	6								
			IND16, Y	37D7	gggg	6								
			IND16, Z	37E7	gggg	6								
			EXT	37F7	hh ll	6								
			E, X	2787	—	6								
			E, Y	2797	—	6								
			E, Z	27A7	—	6								
ORE	OR E	(E) ✣ (M : M + 1) ⇒ E	IMM16	3737	jj kk	4	—	—	—	—	Δ	Δ	0	—
			IND16, X	3747	gggg	6								
			IND16, Y	3757	gggg	6								
			IND16, Z	3767	gggg	6								
			EXT	3777	hh ll	6								
ORP	OR Condition Code Register	(CCR) ✣ IMM16 ⇒ CCR	IMM16	373B	jj kk	4	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ
PSHA	Push A	(SK : SP) + $0001 ⇒ SK : SP Push (A) (SK : SP) – $0002 ⇒ SK : SP	INH	3708	—	4	—	—	—	—	—	—	—	—
PSHB	Push B	(SK : SP) + $0001 ⇒ SK : SP Push (B) (SK : SP) – $0002 ⇒ SK : SP	INH	3718	—	4	—	—	—	—	—	—	—	—
PSHM	Push Multiple Registers Mask bits: 0 = D 1 = E 2 = IX 3 = IY 4 = IZ 5 = K 6 = CCR 7 = (Reserved)	For mask bits 0 to 7: If mask bit set Push register (SK : SP) – 2 ⇒ SK : SP	IMM8	34	ii	4 + 2N N = number of registers pushed	—	—	—	—	—	—	—	—
PSHMAC	Push MAC Registers	MAC Registers ⇒ Stack	INH	27B8	—	14	—	—	—	—	—	—	—	—
PULA	Pull A	(SK : SP) + $0002 ⇒ SK : SP Pull (A) (SK : SP) – $0001 ⇒ SK : SP	INH	3709	—	6	—	—	—	—	—	—	—	—
PULB	Pull B	(SK : SP) + $0002 ⇒ SK : SP Pull (B) (SK : SP) – $0001 ⇒ SK : SP	INH	3719	—	6	—	—	—	—	—	—	—	—

Table 4-2 Instruction Set Summary (Continued)

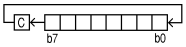
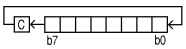
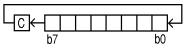
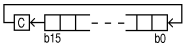
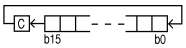
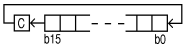
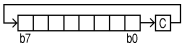
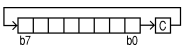
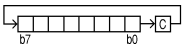
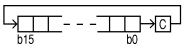
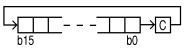
Mnemonic	Operation	Description	Address	Instruction			Condition Codes							
			Mode	Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
PULM	Pull Multiple Registers Mask bits: 0 = CCR[15:4] 1 = K 2 = IZ 3 = IY 4 = IX 5 = E 6 = D 7 = (Reserved)	For mask bits 0 to 7: If mask bit set (SK : SP) + 2 ⇒ SK : SP Pull register	IMM8	35	ii	4+2(N+1) N = number of registers pulled	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ
PULMAC	Pull MAC State	Stack ⇒ MAC Registers	INH	27B9	—	16	—	—	—	—	—	—	—	—
RMAC	Repeating Multiply and Accumulate Signed 16-Bit Fractions	Repeat until (E) < 0 (AM) + (H) ∗ (I) ⇒ AM Qualified (IX) ⇒ IX; Qualified (IY) ⇒ IY; $(M : M + 1)_X \Rightarrow H$; $(M : M + 1)_Y \Rightarrow I$ (E) – 1 ⇒ E Until (E) < $0000	IMM8	FB	xoyo	6 + 12 per iteration	—	Δ	—	Δ	—	—	—	—
ROL	Rotate Left		IND8, X IND8, Y IND8, Z IND16, X IND16, Y IND16, Z EXT	0C 1C 2C 170C 171C 172C 173C	ff ff ff gggg gggg gggg hh ll	8 8 8 8 8 8 8	—	—	—	—	Δ	Δ	Δ	Δ
ROLA	Rotate Left A		INH	370C	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ROLB	Rotate Left B		INH	371C	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ROLD	Rotate Left D		INH	27FC	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ROLE	Rotate Left E		INH	277C	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ROLW	Rotate Left Word		IND16, X IND16, Y IND16, Z EXT	270C 271C 272C 273C	gggg gggg gggg hh ll	8 8 8 8	—	—	—	—	Δ	Δ	Δ	Δ
ROR	Rotate Right Byte		IND8, X IND8, Y IND8, Z IND16, X IND16, Y IND16, Z EXT	0E 1E 2E 170E 171E 172E 173E	ff ff ff gggg gggg gggg hh ll	8 8 8 8 8 8 8	—	—	—	—	Δ	Δ	Δ	Δ
RORA	Rotate Right A		INH	370E	—	2	—	—	—	—	Δ	Δ	Δ	Δ
RORB	Rotate Right B		INH	371E	—	2	—	—	—	—	Δ	Δ	Δ	Δ
RORD	Rotate Right D		INH	27FE	—	2	—	—	—	—	Δ	Δ	Δ	Δ
RORE	Rotate Right E		INH	277E	—	2	—	—	—	—	Δ	Δ	Δ	Δ

Table 4-2 Instruction Set Summary (Continued)

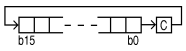
Mnemonic	Operation	Description	Address	Instruction			Condition Codes							
			Mode	Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
RORW	Rotate Right Word		IND16, X IND16, Y IND16, Z EXT	270E 271E 272E 273E	gggg gggg gggg hh ll	8 8 8 8	—	—	—	—	Δ	Δ	Δ	Δ
RTI[3]	Return from Interrupt	(SK : SP) + 2 ⇒ SK : SP Pull CCR (SK : SP) + 2 ⇒ SK : SP Pull PC (PK : PC) – 6 ⇒ PK : PC	INH	2777	—	12	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ
RTS[4]	Return from Subroutine	(SK : SP) + 2 ⇒ SK : SP Pull PK (SK : SP) + 2 ⇒ SK : SP Pull PC (PK : PC) – 2 ⇒ PK : PC	INH	27F7	—	12	—	—	—	—	—	—	—	—
SBA	Subtract B from A	(A) – (B) ⇒ A	INH	370A	—	2	—	—	—	—	Δ	Δ	Δ	Δ
SBCA	Subtract with Carry from A	(A) – (M) – C ⇒ A	IND8, X IND8, Y IND8, Z IMM8 IND16, X IND16, Y IND16, Z EXT E, X E, Y E, Z	42 52 62 72 1742 1752 1762 1772 2742 2752 2762	ff ff ff ii gggg gggg gggg hh ll — — —	6 6 6 2 6 6 6 6 6 6 6	—	—	—	—	Δ	Δ	Δ	Δ
SBCB	Subtract with Carry from B	(B) – (M) – C ⇒ B	IND8, X IND8, Y IND8, Z IMM8 IND16, X IND16, Y IND16, Z EXT E, X E, Y E, Z	C2 D2 E2 F2 17C2 17D2 17E2 17F2 27C2 27D2 27E2	ff ff ff ii gggg gggg gggg hh ll — — —	6 6 6 2 6 6 6 6 6 6 6	—	—	—	—	Δ	Δ	Δ	Δ
SBCD	Subtract with Carry from D	(D) – (M : M + 1) – C ⇒ D	IND8, X IND8, Y IND8, Z IMM16 IND16, X IND16, Y IND16, Z EXT E, X E, Y E, Z	82 92 A2 37B2 37C2 37D2 37E2 37F2 2782 2792 27A2	ff ff ff jj kk gggg gggg gggg hh ll — — —	6 6 6 4 6 6 6 6 6 6 6	—	—	—	—	Δ	Δ	Δ	Δ
SBCE	Subtract with Carry from E	(E) – (M : M + 1) – C ⇒ E	IMM16 IND16, X IND16, Y IND16, Z EXT	3732 3742 3752 3762 3772	jj kk gggg gggg gggg hh ll	4 6 6 6 6	—	—	—	—	Δ	Δ	Δ	Δ
SDE	Subtract D from E	(E) – (D) ⇒ E	INH	2779	—	2	—	—	—	—	Δ	Δ	Δ	Δ
STAA	Store A	(A) ⇒ M	IND8, X IND8, Y IND8, Z IND16, X IND16, Y IND16, Z EXT E, X E, Y E, Z	4A 5A 6A 174A 175A 176A 177A 274A 275A 276A	ff ff ff gggg gggg gggg hh ll — — —	4 4 4 6 6 6 6 4 4 4	—	—	—	—	Δ	Δ	0	—

Table 4-2 Instruction Set Summary (Continued)

Mnemonic	Operation	Description	Address	Instruction			Condition Codes							
			Mode	Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
STAB	Store B	(B) ⇒ M	IND8, X	CA	ff	4	—	—	—	—	Δ	Δ	0	—
			IND8, Y	DA	ff	4								
			IND8, Z	EA	ff	4								
			IND16, X	17CA	gggg	6								
			IND16, Y	17DA	gggg	6								
			IND16, Z	17EA	gggg	6								
			EXT	17FA	hh ll	6								
			E, X	27CA	—	4								
			E, Y	27DA	—	4								
			E, Z	27EA	—	4								
STD	Store D	(D) ⇒ M : M + 1	IND8, X	8A	ff	4	—	—	—	—	Δ	Δ	0	—
			IND8, Y	9A	ff	4								
			IND8, Z	AA	ff	4								
			IND16, X	37CA	gggg	6								
			IND16, Y	37DA	gggg	6								
			IND16, Z	37EA	gggg	6								
			EXT	37FA	hh ll	6								
			E, X	278A	—	6								
			E, Y	279A	—	6								
			E, Z	27AA	—	6								
STE	Store E	(E) ⇒ M : M + 1	IND16, X	374A	gggg	6	—	—	—	—	Δ	Δ	0	—
			IND16, Y	375A	gggg	6								
			IND16, Z	376A	gggg	6								
			EXT	377A	hh ll	6								
STED	Store Concatenated D and E	(E) ⇒ M : M + 1 (D) ⇒ M + 2 : M + 3	EXT	2773	hh ll	8	—	—	—	—	—	—	—	—
STS	Store Stack Pointer	(SP) ⇒ M : M + 1	IND8, X	8F	ff	4	—	—	—	—	Δ	Δ	0	—
			IND8, Y	9F	ff	4								
			IND8, Z	AF	ff	4								
			IND16, X	178F	gggg	6								
			IND16, Y	179F	gggg	6								
			IND16, Z	17AF	gggg	6								
			EXT	17BF	hh ll	6								
STX	Store IX	(IX) ⇒ M : M + 1	IND8, X	8C	ff	4	—	—	—	—	Δ	Δ	0	—
			IND8, Y	9C	ff	4								
			IND8, Z	AC	ff	4								
			IND16, X	178C	gggg	6								
			IND16, Y	179C	gggg	6								
			IND16, Z	17AC	gggg	6								
			EXT	17BC	hh ll	6								
STY	Store IY	(IY) ⇒ M : M + 1	IND8, X	8D	ff	4	—	—	—	—	Δ	Δ	0	—
			IND8, Y	9D	ff	4								
			IND8, Z	AD	ff	4								
			IND16, X	178D	gggg	6								
			IND16, Y	179D	gggg	6								
			IND16, Z	17AD	gggg	6								
			EXT	17BD	hh ll	6								
STZ	Store Z	(IZ) ⇒ M : M + 1	IND8, X	8E	ff	4	—	—	—	—	Δ	Δ	0	—
			IND8, Y	9E	ff	4								
			IND8, Z	AE	ff	4								
			IND16, X	178E	gggg	6								
			IND16, Y	179E	gggg	6								
			IND16, Z	17AE	gggg	6								
			EXT	17BE	hh ll	6								
SUBA	Subtract from A	(A) – (M) ⇒ A	IND8, X	40	ff	6	—	—	—	—	Δ	Δ	Δ	Δ
			IND8, Y	50	ff	6								
			IND8, Z	60	ff	6								
			IMM8	70	ii	2								
			IND16, X	1740	gggg	6								
			IND16, Y	1750	gggg	6								
			IND16, Z	1760	gggg	6								
			EXT	1770	hh ll	6								
			E, X	2740	—	6								
			E, Y	2750	—	6								
			E, Z	2760	—	6								

Table 4-2 Instruction Set Summary (Continued)

Mnemonic	Operation	Description	Address	Instruction			Condition Codes							
			Mode	Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
SUBB	Subtract from B	(B) – (M) ⇒ B	IND8, X	C0	ff	6	—	—	—	—	Δ	Δ	Δ	Δ
			IND8, Y	D0	ff	6								
			IND8, Z	E0	ff	6								
			IMM8	F0	ii	2								
			IND16, X	17C0	gggg	6								
			IND16, Y	17D0	gggg	6								
			IND16, Z	17E0	gggg	6								
			EXT	17F0	hh ll	6								
			E, X	27C0	—	6								
			E, Y	27D0	—	6								
			E, Z	27E0	—	6								
SUBD	Subtract from D	(D) – (M : M + 1) ⇒ D	IND8, X	80	ff	6	—	—	—	—	Δ	Δ	Δ	Δ
			IND8, Y	90	ff	6								
			IND8, Z	A0	ff	6								
			IMM16	37B0	jj kk	4								
			IND16, X	37C0	gggg	6								
			IND16, Y	37D0	gggg	6								
			IND16, Z	37E0	gggg	6								
			EXT	37F0	hh ll	6								
			E, X	2780	—	6								
			E, Y	2790	—	6								
			E, Z	27A0	—	6								
SUBE	Subtract from E	(E) – (M : M + 1) ⇒ E	IMM16	3730	jj kk	4	—	—	—	—	Δ	Δ	Δ	Δ
			IND16, X	3740	gggg	6								
			IND16, Y	3750	gggg	6								
			IND16, Z	3760	gggg	6								
			EXT	3770	hh ll	6								
SWI	Software Interrupt	(PK : PC) + $0002 ⇒ PK : PC Push (PC) (SK : SP) – $0002 ⇒ SK : SP Push (CCR) (SK : SP) – $0002 ⇒ SK : SP $0 ⇒ PK SWI Vector ⇒ PC	INH	3720	—	16	—	—	—	—	—	—	—	—
SXT	Sign Extend B into A	If B7 = 1 then $FF ⇒ A else $00 ⇒ A	INH	27F8	—	2	—	—	—	—	Δ	Δ	—	—
TAB	Transfer A to B	(A) ⇒ B	INH	3717	—	2	—	—	—	—	Δ	Δ	0	—
TAP	Transfer A to CCR	(A[7:0]) ⇒ CCR[15:8]	INH	37FD	—	4	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ
TBA	Transfer B to A	(B) ⇒ A	INH	3707	—	2	—	—	—	—	Δ	Δ	0	—
TBEK	Transfer B to EK	(B[3:0]) ⇒ EK	INH	27FA	—	2	—	—	—	—	—	—	—	—
TBSK	Transfer B to SK	(B[3:0]) ⇒ SK	INH	379F	—	2	—	—	—	—	—	—	—	—
TBXK	Transfer B to XK	(B[3:0]) ⇒ XK	INH	379C	—	2	—	—	—	—	—	—	—	—
TBYK	Transfer B to YK	(B[3:0]) ⇒ YK	INH	379D	—	2	—	—	—	—	—	—	—	—
TBZK	Transfer B to ZK	(B[3:0]) ⇒ ZK	INH	379E	—	2	—	—	—	—	—	—	—	—
TDE	Transfer D to E	(D) ⇒ E	INH	277B	—	2	—	—	—	—	Δ	Δ	0	—
TDMSK	Transfer D to XMSK : YMSK	(D[15:8]) ⇒ X MASK (D[7:0]) ⇒ Y MASK	INH	372F	—	2	—	—	—	—	—	—	—	—
TDP	Transfer D to CCR	(D) ⇒ CCR[15:4]	INH	372D	—	4	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ
TED	Transfer E to D	(E) ⇒ D	INH	27FB	—	2	—	—	—	—	Δ	Δ	0	—
TEDM	Transfer E and D to AM[31:0] Sign Extend AM	(E) ⇒ AM[31:16] (D) ⇒ AM[15:0] AM[35:32] = AM31	INH	27B1	—	4	—	0	—	0	—	—	—	—
TEKB	Transfer EK to B	(EK) ⇒ B[3:0] $0 ⇒ B[7:4]	INH	27BB	—	2	—	—	—	—	—	—	—	—
TEM	Transfer E to AM[31:16] Sign Extend AM Clear AM LSB	(E) ⇒ AM[31:16] $00 ⇒ AM[15:0] AM[35:32] = AM31	INH	27B2	—	4	—	0	—	0	—	—	—	—
TMER	Transfer Rounded AM to E	Rounded (AM) ⇒ Temp If (SM • (EV ✣ MV)) then Saturation Value ⇒ E else Temp[31:16] ⇒ E	INH	27B4	—	6	—	Δ	—	Δ	Δ	Δ	—	—

Table 4-2 Instruction Set Summary (Continued)

Mnemonic	Operation	Description	Address Mode	Instruction Opcode	Instruction Operand	Instruction Cycles	Condition Codes S	MV	H	EV	N	Z	V	C
TMET	Transfer Truncated AM to E	If (SM • (EV ✣ MV)) then Saturation Value ⇒ E else AM[31:16] ⇒ E	INH	27B5	—	2	—	—	—	—	Δ	Δ	—	—
TMXED	Transfer AM to IX : E : D	AM[35:32] ⇒ IX[3:0] AM35 ⇒ IX[15:4] AM[31:16] ⇒ E AM[15:0] ⇒ D	INH	27B3	—	6	—	—	—	—	—	—	—	—
TPA	Transfer CCR to A	(CCR[15:8]) ⇒ A	INH	37FC	—	2	—	—	—	—	—	—	—	—
TPD	Transfer CCR to D	(CCR) ⇒ D	INH	372C	—	2	—	—	—	—	—	—	—	—
TSKB	Transfer SK to B	(SK) ⇒ B[3:0] $0 ⇒ B[7:4]	INH	37AF	—	2	—	—	—	—	—	—	—	—
TST	Test Byte Zero or Minus	(M) – $00	IND8, X IND8, Y IND8, Z IND16, X IND16, Y IND16, Z EXT	06 16 26 1706 1716 1726 1736	ff ff ff gggg gggg gggg hh ll	6 6 6 6 6 6 6	—	—	—	—	Δ	Δ	0	0
TSTA	Test A for Zero or Minus	(A) – $00	INH	3706	—	2	—	—	—	—	Δ	Δ	0	0
TSTB	Test B for Zero or Minus	(B) – $00	INH	3716	—	2	—	—	—	—	Δ	Δ	0	0
TSTD	Test D for Zero or Minus	(D) – $0000	INH	27F6	—	2	—	—	—	—	Δ	Δ	0	0
TSTE	Test E for Zero or Minus	(E) – $0000	INH	2776	—	2	—	—	—	—	Δ	Δ	0	0
TSTW	Test for Zero or Minus Word	(M : M + 1) – $0000	IND16, X IND16, Y IND16, Z EXT	2706 2716 2726 2736	gggg gggg gggg hh ll	6 6 6 6	—	—	—	—	Δ	Δ	0	0
TSX	Transfer SP to X	(SK : SP) + $0002 ⇒ XK : IX	INH	274F	—	2	—	—	—	—	—	—	—	—
TSY	Transfer SP to Y	(SK : SP) + $0002 ⇒ YK : IY	INH	275F	—	2	—	—	—	—	—	—	—	—
TSZ	Transfer SP to Z	(SK : SP) + $0002 ⇒ ZK : IZ	INH	276F	—	2	—	—	—	—	—	—	—	—
TXKB	Transfer XK to B	(XK) ⇒ B[3:0] $0 ⇒ B[7:4]	INH	37AC	—	2	—	—	—	—	—	—	—	—
TXS	Transfer X to SP	(XK : IX) – $0002 ⇒ SK : SP	INH	374E	—	2	—	—	—	—	—	—	—	—
TXY	Transfer X to Y	(XK : IX) ⇒ YK : IY	INH	275C	—	2	—	—	—	—	—	—	—	—
TXZ	Transfer X to Z	(XK : IX) ⇒ ZK : IZ	INH	276C	—	2	—	—	—	—	—	—	—	—
TYKB	Transfer YK to B	(YK) ⇒ B[3:0] $0 ⇒ B[7:4]	INH	37AD	—	2	—	—	—	—	—	—	—	—
TYS	Transfer Y to SP	(YK : IY) – $0002 ⇒ SK : SP	INH	375E	—	2	—	—	—	—	—	—	—	—
TYX	Transfer Y to X	(YK : IY) ⇒ XK : IX	INH	274D	—	2	—	—	—	—	—	—	—	—
TYZ	Transfer Y to Z	(YK : IY) ⇒ ZK : IZ	INH	276D	—	2	—	—	—	—	—	—	—	—
TZKB	Transfer ZK to B	(ZK) ⇒ B[3:0] $0 ⇒ B[7:4]	INH	37AE	—	2	—	—	—	—	—	—	—	—
TZS	Transfer Z to SP	(ZK : IZ) – $0002 ⇒ SK : SP	INH	376E	—	2	—	—	—	—	—	—	—	—
TZX	Transfer Z to X	(ZK : IZ) ⇒ XK : IX	INH	274E	—	2	—	—	—	—	—	—	—	—
TZY	Transfer Z to Y	(ZK : IZ) ⇒ YK : IY	INH	275E	—	2	—	—	—	—	—	—	—	—
WAI	Wait for Interrupt	WAIT	INH	27F3	—	8	—	—	—	—	—	—	—	—
XGAB	Exchange A with B	(A) ⇔ (B)	INH	371A	—	2	—	—	—	—	—	—	—	—
XGDE	Exchange D with E	(D) ⇔ (E)	INH	277A	—	2	—	—	—	—	—	—	—	—
XGDX	Exchange D with IX	(D) ⇔ (IX)	INH	37CC	—	2	—	—	—	—	—	—	—	—

Table 4-2 Instruction Set Summary (Continued)

Mnemonic	Operation	Description	Address	Instruction			Condition Codes							
			Mode	Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
XGDY	Exchange D with IY	(D) ⇔ (IY)	INH	37DC	—	2	—	—	—	—	—	—	—	—
XGDZ	Exchange D with IZ	(D) ⇔ (IZ)	INH	37EC	—	2	—	—	—	—	—	—	—	—
XGEX	Exchange E with IX	(E) ⇔ (IX)	INH	374C	—	2	—	—	—	—	—	—	—	—
XGEY	Exchange E with IY	(E) ⇔ (IY)	INH	375C	—	2	—	—	—	—	—	—	—	—
XGEZ	Exchange E with IZ	(E) ⇔ (IZ)	INH	376C	—	2	—	—	—	—	—	—	—	—

NOTES:

1. CCR[15:4] change according to the results of the operation. The PK field is not affected.
2. Cycle times for conditional branches are shown in “taken, not taken” order.
3. CCR[15:0] change according to the copy of the CCR pulled from the stack.
4. PK field changes according to the state pulled from the stack. The rest of the CCR is not affected.

Table 4-3 Instruction Set Abbreviations and Symbols

A	—	Accumulator A	X	—	Register used in operation
AM	—	Accumulator M	M	—	Address of one memory byte
B	—	Accumulator B	M +1	—	Address of byte at M + $0001
CCR	—	Condition code register	M : M + 1	—	Address of one memory word
D	—	Accumulator D	(…)X	—	Contents of address pointed to by IX
E	—	Accumulator E	(...)Y	—	Contents of address pointed to by IY
EK	—	Extended addressing extension field	(...)Z	—	Contents of address pointed to by IZ
IR	—	MAC multiplicand register	E, X	—	IX with E offset
HR	—	MAC multiplier register	E, Y	—	IY with E offset
IX	—	Index register X	E, Z	—	IZ with E offset
IY	—	Index register Y	EXT	—	Extended
IZ	—	Index register Z	EXT20	—	20-bit extended
K	—	Address extension register	IMM8	—	8-bit immediate
PC	—	Program counter	IMM16	—	16-bit immediate
PK	—	Program counter extension field	IND8, X	—	IX with unsigned 8-bit offset
SK	—	Stack pointer extension field	IND8, Y	—	IY with unsigned 8-bit offset
SL	—	Multiply and accumulate sign latch	IND8, Z	—	IZ with unsigned 8-bit offset
SP	—	Stack pointer	IND16, X	—	IX with signed 16-bit offset
XK	—	Index register X extension field	IND16, Y	—	IY with signed 16-bit offset
YK	—	Index register Y extension field	IND16, Z	—	IZ with signed 16-bit offset
ZK	—	Index register Z extension field	IND20, X	—	IX with signed 20-bit offset
XMSK	—	Modulo addressing index register X mask	IND20, Y	—	IY with signed 20-bit offset
YMSK	—	Modulo addressing index register Y mask	IND20, Z	—	IZ with signed 20-bit offset
S	—	Stop disable control bit	INH	—	Inherent
MV	—	AM overflow indicator	IXP	—	Post-modified indexed
H	—	Half carry indicator	REL8	—	8-bit relative
EV	—	AM extended overflow indicator	REL16	—	16-bit relative
N	—	Negative indicator	b	—	4-bit address extension
Z	—	Zero indicator	ff	—	8-bit unsigned offset
V	—	Two's complement overflow indicator	gggg	—	16-bit signed offset
C	—	Carry/borrow indicator	hh	—	High byte of 16-bit extended address
IP	—	Interrupt priority field	ii	—	8-bit immediate data
SM	—	Saturation mode control bit	jj	—	High byte of 16-bit immediate data
PK	—	Program counter extension field	kk	—	Low byte of 16-bit immediate data
—	—	Bit not affected	ll	—	Low byte of 16-bit extended address
Δ	—	Bit changes as specified	mm	—	8-bit mask
0	—	Bit cleared	mmmm	—	16-bit mask
1	—	Bit set	rr	—	8-bit unsigned relative offset
M	—	Memory location used in operation	rrrr	—	16-bit signed relative offset
R	—	Result of operation	xo	—	MAC index register X offset
S	—	Source data	yo	—	MAC index register Y offset
			z	—	4-bit zero extension

+	—	Addition	•	—	AND
−	—	Subtraction or negation (two's complement)	✣	—	Inclusive OR (OR)
*	—	Multiplication	⊕	—	Exclusive OR (EOR)
/	—	Division	$\overline{\text{NOT}}$	—	Complementation
>	—	Greater	:	—	Concatenation
<	—	Less	⇒	—	Transferred
=	—	Equal	⇔	—	Exchanged
≥	—	Equal or greater	±	—	Sign bit; also used to show tolerance
≤	—	Equal or less	«	—	Sign extension
≠	—	Not equal	%	—	Binary value
			$	—	Hexadecimal value

4.8 Comparison of CPU16 and M68HC11 CPU Instruction Sets

Most M68HC11 CPU instructions are a source-code compatible subset of the CPU16 instruction set. However, certain M68HC11 CPU instructions have been replaced by functionally equivalent CPU16 instructions, and some CPU16 instructions with the same mnemonics as M68HC11 CPU instructions operate differently.

Table 4-4 shows M68HC11 CPU instructions that either have been replaced by CPU16 instructions or that operate differently on the CPU16. Replacement instructions are not identical to M68HC11 CPU instructions. M68HC11 code must be altered to establish proper preconditions.

All CPU16 instruction execution times differ from those of the M68HC11. *Motorola Programming Note* M68HC16PN01/D, *Transporting M68HC11 Code to M68HC16 Devices*, contains detailed information about differences between the two instruction sets. Refer to the *CPU16 Reference Manual* (CPU16RM/AD) for further details about CPU operations.

Table 4-4 CPU16 Implementation of M68HC11 CPU Instructions

M68HC11 Instruction	CPU16 Implementation
BHS	BCC only
BLO	BCS only
BSR	Generates a different stack frame
CLC	Replaced by ANDP
CLI	Replaced by ANDP
CLV	Replaced by ANDP
DES	Replaced by AIS
DEX	Replaced by AIX
DEY	Replaced by AIY
INS	Replaced by AIS
INX	Replaced by AIX
INY	Replaced by AIY
JMP	IND8 and EXT addressing modes replaced by IND20 and EXT20 modes
JSR	IND8 and EXT addressing modes replaced by IND20 and EXT20 modes Generates a different stack frame
LSL, LSLD	Use ASL instructions[1]
PSHX	Replaced by PSHM
PSHY	Replaced by PSHM
PULX	Replaced by PULM
PULY	Replaced by PULM
RTI	Reloads PC and CCR only
RTS	Uses two-word stack frame
SEC	Replaced by ORP
SEI	Replaced by ORP
SEV	Replaced by ORP
STOP	Replaced by LPSTOP
TAP	CPU16 CCR bits differ from M68HC11 CPU16 interrupt priority scheme differs from M68HC11
TPA	CPU16 CCR bits differ from M68HC11 CPU16 interrupt priority scheme differs from M68HC11
TSX	Adds 2 to SK : SP before transfer to XK : IX
TSY	Adds 2 to SK : SP before transfer to YK : IY
TXS	Subtracts 2 from XK : IX before transfer to SK : SP
TXY	Transfers XK field to YK field
TYS	Subtracts 2 from YK : IY before transfer to SK : SP
TYX	Transfers YK field to XK field
WAI	Waits indefinitely for interrupt or reset Generates a different stack frame

NOTES:

1. Motorola assemblers automatically translate ASL mnemonics.

4.9 Instruction Format

CPU16 instructions consist of an 8-bit opcode that can be preceded by an 8-bit prebyte and followed by one or more operands.

Opcodes are mapped in four 256-instruction pages. Page 0 opcodes stand alone. Page 1, 2, and 3 opcodes are pointed to by a prebyte code on page 0. The prebytes are $17 (page 1), $27 (page 2), and $37 (page 3).

Operands can be four bits, eight bits or sixteen bits in length. Since the CPU16 fetches 16-bit instruction words from even-byte boundaries, each instruction must contain an even number of bytes.

Operands are organized as bytes, words, or a combination of bytes and words. Operands of four bits are either zero-extended to eight bits, or packed two to a byte. The largest instructions are six bytes in length. Size, order, and function of operands are evaluated when an instruction is decoded.

A page 0 opcode and an 8-bit operand can be fetched simultaneously. Instructions that use 8-bit indexed, immediate, and relative addressing modes have this form. Code written with these instructions is very compact.

Figure 4-4 shows basic CPU16 instruction formats.

8-Bit Opcode with 8-Bit Operand

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Opcode								Operand							

8-Bit Opcode with 4-Bit Index Extensions

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Opcode								X Extension				Y Extension			

8-Bit Opcode, Argument(s)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Opcode								Operand							
Operand(s)															
Operand(s)															

8-Bit Opcode with 8-Bit Prebyte, No Argument

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Prebyte								Opcode							

8-Bit Opcode with 8-Bit Prebyte, Argument(s)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Prebyte								Opcode							
Operand(s)															
Operand(s)															

8-Bit Opcode with 20-Bit Argument

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Opcode								$0				Extension			
Operand															

Figure 4-4 Basic Instruction Formats

4.10 Execution Model

This description builds up a conceptual model of the mechanism the CPU16 uses to fetch and execute instructions. The functional divisions in the model do not necessarily correspond to physical subunits of the microprocessor.

As shown in **Figure 4-5**, there are three functional blocks involved in fetching, decoding, and executing instructions. These are the microsequencer, the instruction pipeline, and the execution unit. These elements function concurrently. All three may be active at any given time.

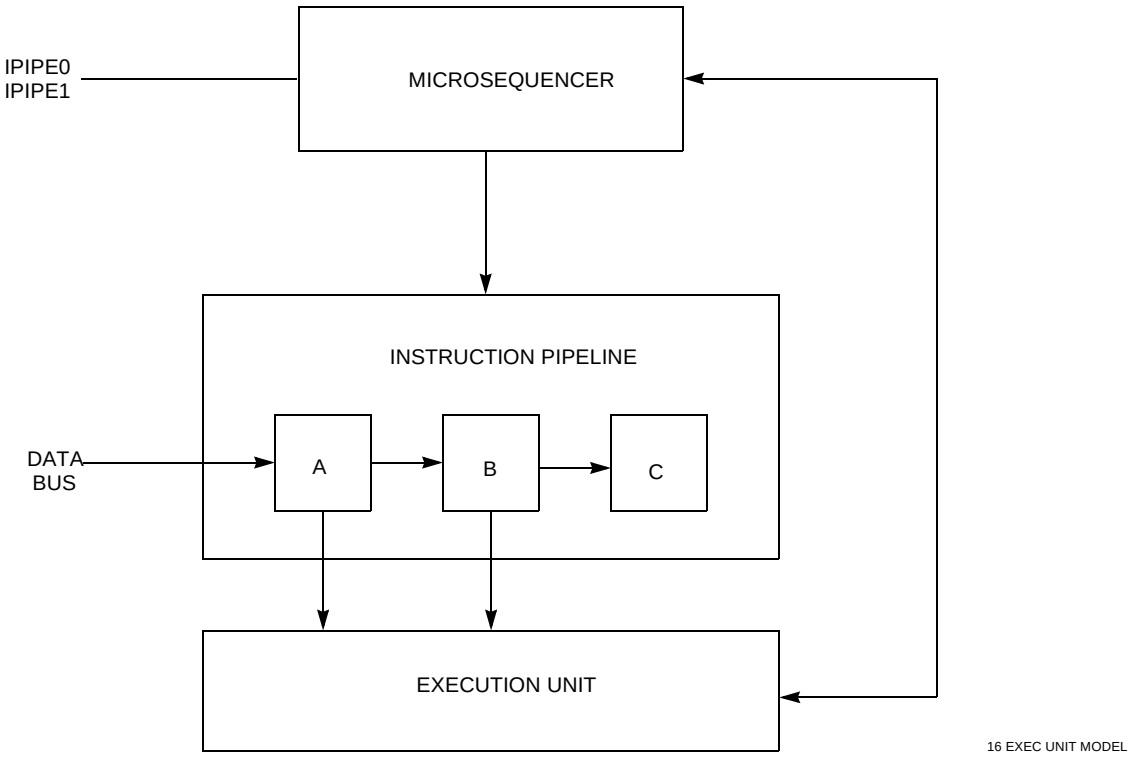


Figure 4-5 Instruction Execution Model

4.10.1 Microsequencer

The microsequencer controls the order in which instructions are fetched, advanced through the pipeline, and executed. It increments the program counter and generates multiplexed external tracking signals IPIPE0 and IPIPE1 from internal signals that control execution sequence.

4.10.2 Instruction Pipeline

The pipeline is a three stage first in, first out buffer (FIFO) that holds instructions while they are decoded and executed. Depending upon instruction size, as many as three instructions can be in the pipeline at one time (single-word instructions, one held in stage C, one being executed in stage B, and one latched in stage A).

4.10.3 Execution Unit

The execution unit evaluates opcodes, interfaces with the microsequencer to advance instructions through the pipeline, and performs instruction operations.

4.11 Execution Process

Fetched opcodes are latched into stage A, then advanced to stage B. Opcodes are evaluated in stage B. The execution unit can access operands in either stage A or stage B (stage B accesses are limited to 8-bit operands). When execution is complete, opcodes are moved from stage B to stage C, where they remain until the next instruction is complete.

A prefetch mechanism in the microsequencer reads instruction words from memory and increments the program counter. When instruction execution begins, the program counter points to an address six bytes after the address of the first word of the instruction being executed.

The number of machine cycles necessary to complete an execution sequence varies according to the complexity of the instruction. Refer to the *CPU16 Reference Manual* (CPU16RM/AD) for details.

4.11.1 Changes in Program Flow

When program flow changes, instructions are fetched from a new address. Before execution can begin at the new address, instructions and operands from the previous instruction stream must be removed from the pipeline. If a change in flow is temporary, a return address must be stored, so that execution of the original instruction stream can resume after the change in flow.

When an instruction that causes a change in program flow executes, PK:PC point to the address of the first word of the instruction + $0006. During execution of the instruction, PK:PC is loaded with the address of the first instruction word in the new instruction stream. However, stages A and B still contain words from the old instruction stream. Extra processing steps must be performed before execution from the new instruction stream.

4.12 Instruction Timing

The execution time of CPU16 instructions has three components:

- Bus cycles required to prefetch the next instruction
- Bus cycles required for operand accesses
- Time required for internal operations

A bus cycle requires a minimum of two system clock periods. If the access time of a memory device is greater than two clock periods, bus cycles are longer. However, all bus cycles must be an integer number of clock periods. CPU16 internal operations are always an integer multiple of two clock periods.

Dynamic bus sizing affects bus cycle time. The integration module manages all accesses. Refer to SECTION 5 SINGLE-CHIP INTEGRATION MODULE 2 for more information.

The CPU16 does not execute more than one instruction at a time. The total time required to execute a particular instruction stream can be calculated by summing the individual execution times of each instruction in the stream.

Total execution time is calculated using the expression:

$$(CL_T) = (CL_P) + (CL_O) + (CL_I)$$

Where:

(CL_T) = Total clock periods per instruction

(CL_I) = Clock periods used for internal operation

(CL_P) = Clock periods used for program access

(CL_O) = Clock periods used for operand access

Refer to the *CPU16 Reference Manual* (CPU16RM/AD) for more information on this topic.

4.13 Exceptions

An exception is an event that preempts normal instruction processing. Exception processing makes the transition from normal instruction execution to execution of a routine that deals with the exception.

Each exception has an assigned vector that points to an associated handler routine. Exception processing includes all operations required to transfer control to a handler routine, but does not include execution of the handler routine itself. Keep the distinction between exception processing and execution of an exception handler in mind while reading this section.

4.13.1 Exception Vectors

An exception vector is the address of a routine that handles an exception. Exception vectors are contained in a data structure called the exception vector table, which is located in the first 512 bytes of bank 0. Refer to **Table 4-5** for the exception vector table.

All vectors except the reset vector consist of one word and reside in data space. The reset vector consists of four words that reside in program space. Refer to **SECTION 5 SINGLE-CHIP INTEGRATION MODULE 2** for information concerning address space types and the function code outputs. There are 52 predefined or reserved vectors, and 200 user-defined vectors.

Each vector is assigned an 8-bit number. Vector numbers for some exceptions are generated by external devices; others are supplied by the processor. There is a direct mapping of vector number to vector table address. The processor left shifts the vector number one place (multiplies by two) to convert it to an address.

Table 4-5 Exception Vector Table

Vector Number	Vector Address	Address Space	Type of Exception
0	0000	P	Reset — Initial ZK, SK, and PK
	0002	P	Reset — Initial PC
	0004	P	Reset — Initial SP
	0006	P	Reset — Initial IZ (Direct Page)
4	0008	D	Breakpoint
5	000A	D	Bus Error
6	000C	D	Software Interrupt
7	000E	D	Illegal Instruction
8	0010	D	Division by Zero
9 – E	0012 – 001C	D	Unassigned, Reserved
F	001E	D	Uninitialized Interrupt
10	0020	D	Unassigned, Reserved
11	0022	D	Level 1 Interrupt Autovector
12	0024	D	Level 2 Interrupt Autovector
13	0026	D	Level 3 Interrupt Autovector
14	0028	D	Level 4 Interrupt Autovector
15	002A	D	Level 5 Interrupt Autovector
16	002C	D	Level 6 Interrupt Autovector
17	002E	D	Level 7 Interrupt Autovector
18	0030	D	Spurious Interrupt
19 – 37	0032 – 006E	D	Unassigned, Reserved
38 – FF	0070 – 01FE	D	User-Defined Interrupts

4.13.2 Exception Stack Frame

During exception processing, the contents of the program counter and condition code register are stacked at a location pointed to by SK:SP. Unless it is altered during exception processing, the stacked PK:PC value is the address of the next instruction in the current instruction stream, plus $0006. **Figure 4-6** shows the exception stack frame.

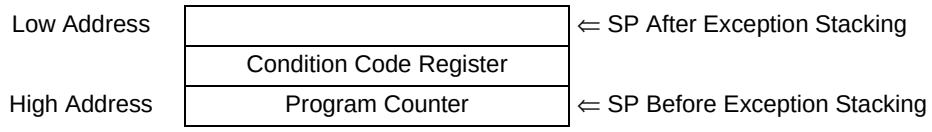


Figure 4-6 Exception Stack Frame Format

4.13.3 Exception Processing Sequence

Exception processing is performed in four phases. Priority of all pending exceptions is evaluated and the highest priority exception is processed first. Processor state is stacked, then the CCR PK extension field is cleared. An exception vector number is acquired and converted to a vector address. The content of the vector address is loaded into the PC and the processor jumps to the exception handler routine.

There are variations within each phase for differing types of exceptions. However, all vectors except RESET are 16-bit addresses, and the PK field is cleared during exception processing. Consequently, exception handlers must be located within bank 0 or vectors must point to a jump table in bank 0.

4.13.4 Types of Exceptions

Exceptions can be either internally or externally generated. External exceptions, which are defined as asynchronous, include interrupts, bus errors, breakpoints, and resets. Internal exceptions, which are defined as synchronous, include the software interrupt (SWI) instruction, the background (BGND) instruction, illegal instruction exceptions, and the divide-by-zero exception.

4.13.4.1 Asynchronous Exceptions

Asynchronous exceptions occur without reference to CPU16 or IMB clocks, but exception processing is synchronized. For all asynchronous exceptions but RESET, exception processing begins at the first instruction boundary following recognition of an exception. Refer to 5.8.1 Interrupt Exception Processing for more information concerning asynchronous exceptions.

Because of pipelining, the stacked return PK:PC value for all asynchronous exceptions, other than reset, is equal to the address of the next instruction in the current instruction stream plus $0006. The RTI instruction, which must terminate all exception handler routines, subtracts $0006 from the stacked value to resume execution of the interrupted instruction stream.

4.13.4.2 Synchronous Exceptions

Synchronous exception processing is part of an instruction definition. Exception processing for synchronous exceptions is always completed, and the first instruction of the handler routine is always executed, before interrupts are detected.

Because of pipelining, the value of PK:PC at the time a synchronous exception executes is equal to the address of the instruction that causes the exception plus $0006. Because RTI always subtracts $0006 upon return, the stacked PK:PC must be adjusted by the instruction that caused the exception so that execution resumes with the following instruction. For this reason, $0002 is added to the PK:PC value before it is stacked.

4.13.5 Multiple Exceptions

Each exception has a hardware priority based upon its relative importance to system operation. Asynchronous exceptions have higher priorities than synchronous exceptions. Exception processing for multiple exceptions is completed by priority, from highest to lowest. Priority governs the order in which exception processing occurs, not the order in which exception handlers are executed.

Unless a bus error, a breakpoint, or a reset occurs during exception processing, the first instruction of all exception handler routines is guaranteed to execute before another exception is processed. Because interrupt exceptions have higher priority than synchronous exceptions, the first instruction in an interrupt handler are executed before other interrupts are sensed.

Bus error, breakpoint, and reset exceptions that occur during exception processing of a previous exception are processed before the first instruction of that exception's handler routine. The converse is not true. If an interrupt occurs during bus error exception processing, for example, the first instruction of the exception handler is executed before interrupts are sensed. This permits the exception handler to mask interrupts during execution.

Refer to SECTION 5 SINGLE-CHIP INTEGRATION MODULE 2 for detailed information concerning interrupts and system reset. Refer to the *CPU16 Reference Manual* (CPU16RM/AD) for information concerning processing of specific exceptions.

4.13.6 RTI Instruction

The return-from-interrupt instruction (RTI) must be the last instruction in all exception handlers except the RESET handler. RTI pulls the exception stack frame that was pushed onto the system stack during exception processing, and restores processor state. Normal program flow resumes at the address of the instruction that follows the last instruction executed before exception processing began.

RTI is not used in the RESET handler because RESET initializes the stack pointer and does not create a stack frame.

4.14 Development Support

The CPU16 incorporates powerful tools for tracking program execution and for system debugging. These tools are deterministic opcode tracking, breakpoint exceptions, and background debug mode. Judicious use of CPU16 capabilities permits in-circuit emulation and system debugging using a bus state analyzer, a simple serial interface, and a terminal.

4.14.1 Deterministic Opcode Tracking

The CPU16 has two multiplexed outputs, IPIPE0 and IPIPE1, that enable external hardware to monitor the instruction pipeline during normal program execution. The signals IPIPE0 and IPIPE1 can be demultiplexed into six pipeline state signals that allow a state analyzer to synchronize with instruction stream activity.

4.14.1.1 IPIPE0/IPIPE1 Multiplexing

Six types of information are required to track pipeline activity. To generate the six state signals, eight pipeline states are encoded and multiplexed into IPIPE0 and IPIPE1. The multiplexed signals have two phases. State signals are active low. **Table 4-6** shows the encoding scheme.

Table 4-6 IPIPE0/IPIPE1 Encoding

Phase	IPIPE1 State	IPIPE0 State	State Signal Name
1	0	0	START and FETCH
	0	1	FETCH
	1	0	START
	1	1	NULL
2	0	0	INVALID
	0	1	ADVANCE
	1	0	EXCEPTION
	1	1	NULL

IPIPE0 and IPIPE1 are timed so that a logic analyzer can capture all six pipeline state signals and address, data, or control bus state in any single bus cycle. Refer to APPENDIX A ELECTRICAL CHARACTERISTICS for specifications.

State signals can be latched asynchronously on the falling and rising edges of either address strobe ($\overline{AS}$) or data strobe ($\overline{DS}$). They can also be latched synchronously using the microcontroller CLKOUT signal. Refer to the *CPU16 Reference Manual* (CPU16RM/AD) for more information on the CLKOUT signal, state signals, and state signal demux logic.

4.14.1.2 Combining Opcode Tracking with Other Capabilities

Pipeline state signals are useful during normal instruction execution and execution of exception handlers. The signals provide a complete model of the pipeline up to the point a breakpoint is acknowledged.

Breakpoints are acknowledged after an instruction has executed, when it is in pipeline Stage C. A breakpoint can initiate either exception processing or background debugging mode. IPIPE0/IPIPE1 are not usable when the CPU16 is in background debugging mode.

4.14.2 Breakpoints

Breakpoints are set by assertion of the microcontroller $\overline{BKPT}$ pin. The CPU16 supports breakpoints on any memory access. Acknowledged breakpoints can initiate either exception processing or background debug mode. After BDM has been enabled, the CPU16 will enter BDM when the $\overline{BKPT}$ input is asserted.

- If $\overline{BKPT}$ assertion is synchronized with an instruction prefetch, the instruction is tagged with the breakpoint when it enters the pipeline, and the breakpoint occurs after the instruction executes.

- If $\overline{\text{BKPT}}$ assertion is synchronized with an operand fetch, breakpoint processing occurs at the end of the instruction during which $\overline{\text{BKPT}}$ is latched.

Breakpoints on instructions that are flushed from the pipeline before execution are not acknowledged. Operand breakpoints are always acknowledged. There is no breakpoint acknowledge bus cycle when BDM is entered. Refer to 5.6.4.1 Breakpoint Acknowledge Cycle for more information about breakpoints.

4.14.3 Opcode Tracking and Breakpoints

Breakpoints are acknowledged after a tagged instruction has executed, that is when the instruction is copied from pipeline stage B to stage C. Stage C contains the opcode of the previous instruction when execution of the current instruction begins.

When an instruction is tagged, IPIPE0/IPIPE1 reflect the start of execution and the appropriate number of pipeline advances and operand fetches before the breakpoint is acknowledged. If background debug mode is enabled, these signals model the pipeline before BDM is entered.

4.14.4 Background Debug Mode

Microprocessor debugging programs are generally implemented in external software. CPU16 BDM provides a debugger implemented in CPU microcode. BDM incorporates a full set of debug options. Registers can be viewed and altered, memory can be read or written, and test features can be invoked. BDM is an alternate CPU16 operating mode. While the CPU16 is in BDM, normal instruction execution is suspended, and special microcode performs debugging functions under external control. While in BDM, the CPU16 ceases to fetch instructions through the data bus and communicates with the development system through a dedicated serial interface.

4.14.5 Enabling BDM

The CPU16 samples the $\overline{\text{BKPT}}$ input during reset to determine whether to enable BDM. When $\overline{\text{BKPT}}$ is asserted at the rising edge of the $\overline{\text{RESET}}$ signal, BDM operation is enabled. BDM remains enabled until the next system reset. If $\overline{\text{BKPT}}$ is at logic level one on the trailing edge of $\overline{\text{RESET}}$, BDM is disabled. $\overline{\text{BKPT}}$ is relatched on each rising transition of $\overline{\text{RESET}}$. $\overline{\text{BKPT}}$ is synchronized internally and must be asserted for at least two clock cycles before negation of $\overline{\text{RESET}}$.

4.14.5.1 BDM Sources

When BDM is enabled, external breakpoint hardware and the BGND instruction can cause the CPU16 to enter BDM. If BDM is not enabled when a breakpoint occurs, a breakpoint exception is processed.

4.14.5.2 Entering BDM

When the CPU16 detects a breakpoint or decodes a BGND instruction when BDM is enabled, it suspends instruction execution and asserts the FREEZE signal. Once FREEZE has been asserted, the CPU16 enables the BDM serial communication hardware and awaits a command. Assertion of FREEZE causes opcode tracking signals IPIPE0 and IPIPE1 to change definition and become serial communication signals DSO and DSI. FREEZE is asserted at the next instruction boundary after the assertion of $\overline{\text{BKPT}}$ or execution of the BGND instruction. IPIPE0 and IPIPE1 change function before an exception signal can be generated. The development system must use FREEZE assertion as an indication that BDM has been entered. When BDM is exited, FREEZE is negated before initiation of normal bus cycles. IPIPE0 and IPIPE1 are valid when normal instruction prefetch begins.

4.14.5.3 BDM Commands

Commands consist of one 16-bit operation word and can include one or more 16-bit extension words. Each incoming word is read as it is assembled by the serial interface. The microcode routine corresponding to a command is executed as soon as the command is complete. Result operands are loaded into the output shift register to be shifted out as the next command is read. This process is repeated for each command until the CPU returns to normal operating mode. The BDM command set is summarized in **Table 4-7**. Refer to the *CPU16 Reference Manual* (CPU16RM/AD) for a BDM command glossary.

Table 4-7 Command Summary

Command	Mnemonic	Description
Read Registers from Mask	RREGM	Read contents of registers specified by command word register mask
Write Registers from Mask	WREGM	Write to registers specified by command word register mask
Read MAC Registers	RDMAC	Read contents of entire multiply and accumulate register set
Write MAC Registers	WRMAC	Write to entire multiply and accumulate register set
Read PC and SP	RPCSP	Read contents of program counter and stack pointer
Write PC and SP	WPCSP	Write to program counter and stack pointer
Read Data Memory	RDMEM	Read byte from specified 20-bit address in data space
Write Data Memory	WDMEM	Write byte to specified 20-bit address in data space
Read Program Memory	RPMEM	Read word from specified 20-bit address in program space
Write Program Memory	WPMEM	Write word to specified 20-bit address in program space
Execute from Current PK : PC	GO	Instruction pipeline flushed and refilled; instructions executed from current PC – $0006
Null Operation	NOP	Null command performs no operation

4.14.5.4 Returning from BDM

BDM is terminated when a resume execution (GO) command is received. GO refills the instruction pipeline from address (PK:PC - $0006). FREEZE is negated before the first prefetch. Upon negation of FREEZE, the BDM serial subsystem is disabled and the DSO/DSI signals revert to IPIPE0/IPIPE1 functionality.

4.14.5.5 BDM Serial Interface

The BDM serial interface uses a synchronous protocol similar to that of the Motorola serial peripheral interface (SPI). **Figure 4-7** is a diagram of the serial logic required to use BDM with a development system.

The development system serves as the master of the serial link, and is responsible for the generation of the serial interface clock signal (DSCLK).

Serial clock frequency range is from DC to one-half the CPU16 clock frequency. If DSCLK is derived from the CPU16 system clock, development system serial logic can be synchronized with the target processor.

The serial interface operates in full-duplex mode. Data transfers occur on the falling edge of DSCLK and are stable by the following rising edge of DSCLK. Data is transmitted MSB first, and is latched on the rising edge of DSCLK.

The serial data word is 17 bits wide, which includes 16 data bits and a status/control bit. Bit 16 indicates status of CPU-generated messages.

Command and data transfers initiated by the development system must clear bit 16. All commands that return a result return 16 bits of data plus one status bit.

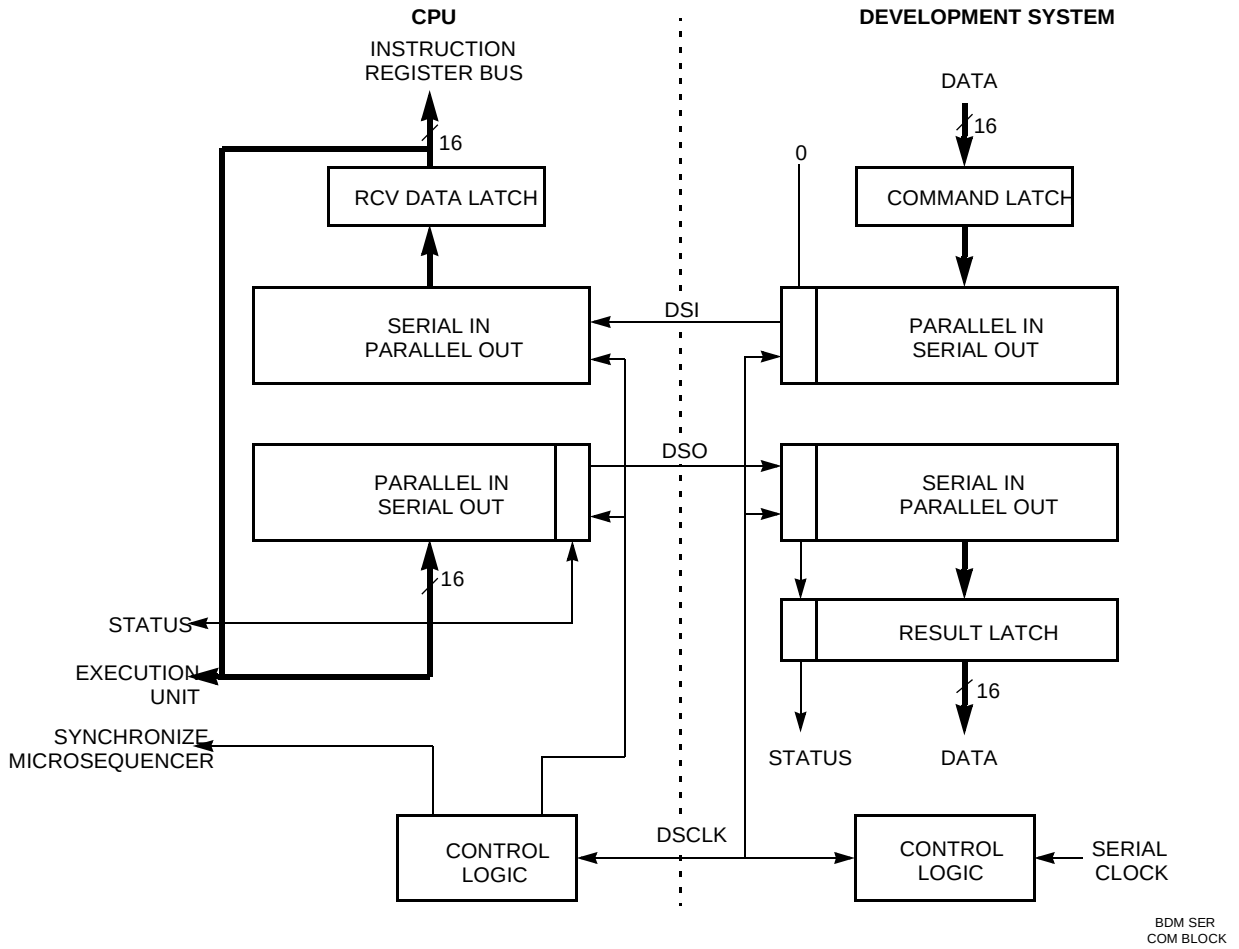


Figure 4-7 BDM Serial I/O Block Diagram

4.15 Recommended BDM Connection

In order to use BDM development tools when an MCU is installed in a system, Motorola recommends that appropriate signal lines be routed to a male Berg connector or double-row header installed on the circuit board with the MCU. Refer to **Figure 4-8**.

Figure 4-8 BDM Connector Pinout

4.16 Digital Signal Processing

The CPU16 performs low-frequency digital signal processing (DSP) algorithms in real time. The most common DSP operation in embedded control applications is filtering, but the CPU16 can perform several other useful DSP functions. These include auto-correlation (detecting a periodic signal in the presence of noise), cross-correlation (determining the presence of a defined periodic signal), and closed-loop control routines (selective filtration in a feedback path).

Although derivation of DSP algorithms is often a complex mathematical task, the algorithms themselves typically consist of a series of multiply and accumulate (MAC) operations. The CPU16 contains a dedicated set of registers that perform MAC operations. As a group, these registers are called the MAC unit.

DSP operations generally require a large number of MAC iterations. The CPU16 instruction set includes instructions that perform MAC setup and repetitive MAC operations. Other instructions, such as 32-bit load and store instructions, can also be used in DSP routines.

Many DSP algorithms require extensive data address manipulation. To increase throughput, the CPU16 performs effective address calculations and data prefetches during MAC operations. In addition, the MAC unit provides modulo addressing to implement circular DSP buffers efficiently.

Refer to the *CPU16 Reference Manual* (CPU16RM/AD) for detailed information concerning the MAC unit and execution of DSP instructions.

SECTION 5
SINGLE-CHIP INTEGRATION MODULE 2

This section is an overview of the single-chip integration module 2 (SCIM2). The SCIM2 is an improved version of the original SCIM and offers the following enhancements:

- Phase-locked loop operation with either a slow (32.768 kHz nominal) or fast (4.194 MHz nominal) reference crystal.
- Reset controller changes that permit slower $\overline{\text{RESET}}$ pin rise times during power-on and external reset sequences.

Refer to the *SCIM Reference Manual* (SCIMRM/AD) for a comprehensive discussion of SCIM capabilities. The differences between the original SCIM and SCIM2 are noted, where necessary, in this chapter. Refer to **APPENDIX D REGISTER SUMMARY** for information concerning the SCIM2 address map and register structure.

5.1 General

The single-chip integration module 2 (SCIM2) consists of six submodules that, with a minimum of external devices, control system startup, initialization, configuration, and the external bus. **Figure 5-1** shows a block diagram of the SCIM2.

The system configuration block controls MCU configuration and operating mode.

The system clock generates clock signals used by the SCIM2, other IMB modules, and external devices. Circuitry is included to detect loss of the phase-locked loop (PLL) reference frequency and to control clock operation in low-power stop mode.

The system protection block provides bus and software watchdog monitors. It also incorporates a periodic interrupt generator that support execution of time-critical control routines.

The external bus interface handles the transfer of information between the CPU16 and external address space. Ports A, B, E, F, G, and H comprise the EBI and may be used for discrete I/O subject to the MCU's operating mode.

The chip-select block provides nine general-purpose chip-select signals and two emulation support chip-select signals. Each general-purpose chip select has associated base address and option registers that control the programmable characteristics of the chip select. Chip-select pins can also be used as general-purpose output port C.

The system test block incorporates hardware necessary for testing the MCU. It is used to perform factory tests, and its use in normal applications is not supported.

The SCIM2 has three basic operating modes:

- 16-bit expanded mode

- 8-bit expanded mode
- Single-chip mode

Operating mode is determined by the logic states of specific MCU pins during reset. Refer to **5.7.8 Operating Configuration Out of Reset** for more detailed information on MCU operating modes.

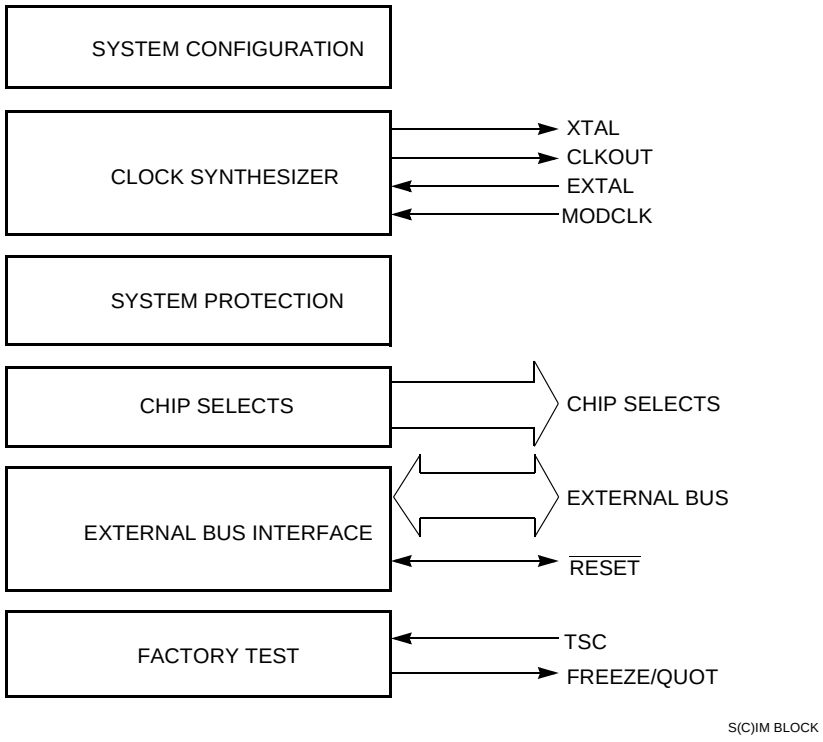


Figure 5-1 SCIM2 Block Diagram

5.2 System Configuration

The MCU can operate as a stand-alone device in single-chip mode, or it can operate with the support of external memory and/or peripheral devices in the 16-bit or 8-bit expanded modes. System configuration is determined by asserting MCU pins during reset and by setting bits in the SCIM2 configuration register (SCIMCR).

5.2.1 Module Mapping

Control registers for all the modules in the microcontroller are mapped into a 4-Kbyte block. The state of the module mapping (MM) bit in SCIMCR determines where the control register block is located in the system memory map. When MM = 0, register addresses range from $7FF000 to $7FFFFF; when MM = 1, register addresses range from $FFF000 to $FFFFFF.

In M68HC16 R-series MCUs, ADDR[23:20] follow the logic state of ADDR19 unless externally driven. MM corresponds to IMB ADDR23. If MM is cleared, the SCIM2 maps IMB modules into address space $7FF000 – $7FFFFF, which is inaccessible to the CPU16. Modules remain inaccessible until reset occurs. The reset state of MM is one, but the bit can be written once. Initialization software should make certain MM remains set to one.

5.2.2 Interrupt Arbitration

Each module that can request interrupts has an interrupt arbitration (IARB) field. Arbitration between interrupt requests of the same priority is performed by serial contention between IARB field bit values. Contention will take place whenever an interrupt request is acknowledged, even when there is only a single request pending. For an interrupt to be serviced, the appropriate IARB field must have a non-zero value. If an interrupt request from a module with an IARB field value of %0000 is recognized, the CPU16 processes a spurious interrupt exception.

Because the SCIM2 routes external interrupt requests to the CPU16, the SCIM2 IARB field value is used for arbitration between internal and external interrupts of the same priority. The reset value of IARB for the SCIM2 is %1111, and the reset IARB value for all other modules is %0000. This prevents SCIM2 interrupts from being discarded during initialization. Refer to **5.8 Interrupts** for a discussion of interrupt arbitration.

5.2.3 Noise Reduction in Single-Chip Mode

Four bits in SCIMCR control pins that can be disabled in single-chip mode to reduce MCU noise emissions. The characteristics of these control bits are listed in **Table 5-1**. Except for EXOFF, these bits disable their associated pins when the MCU is configured for single-chip mode ($\overline{BERR}$ = 0 during reset).

Table 5-1 SCIMCR Noise Control Bits

Bit Mnemonic	Position in SCIMCR	Function	Reset State
EXOFF	15	Disables CLKOUT when set to one.	0
CPUD	12	Disables IPIPE1/DSI and IPIPE0/DSO pins when set to one.	$\overline{BERR}$
ABD	5	Disables ADDR[2:0] when set to one.	
RWD	4	Disables R/$\overline{W}$ when set to one.	

EXOFF disables the CLKOUT external clock output pin by placing it in a high-impedance state. CLKOUT is enabled at power-up, unless explicitly disabled by writing a zero to EXOFF.

CPUD disables the IPIPE1/DSI and IPIPE0/DSO instruction tracking pins by placing them in a high-impedance state when the MCU is not in background debug mode (BDM). When the MCU enters BDM and FREEZE is asserted, IPIPE1/DSI and IPIPE0/DSO become active and serve as the BDM serial I/O lines.

ABD and RWD disable the ADDR[2:0] and R/$\overline{W}$ pins by placing them in a high-impedance state. These pins should be disabled because they cannot be used for discrete I/O and have no use in single-chip mode.

5.2.4 Show Internal Cycles

A show cycle allows internal bus transfers to be monitored externally. The SHEN field in SCIMCR determines what the external bus interface does during internal transfer operations. **Table 5-2** shows whether data is driven externally, and whether external bus arbitration can occur. Refer to **5.6.6.1 Show Cycles** for more information.

Table 5-2 Show Cycle Enable Bits

SHEN[1:0]	Effect
00	Show cycles disabled, external arbitration enabled
01	Show cycles enabled, external arbitration disabled
10	Show cycles enabled, external arbitration enabled
11	Show cycles enabled, external arbitration enabled; internal activity is halted by a bus grant

5.2.5 FREEZE Assertion Response

When the CPU16 enters background debug mode, it asserts the IMB FREEZE signal. The FRZ[1:0] bits in SCIMCR control the behavior of the software watchdog, periodic interrupt timer, and bus monitor in response to FREEZE assertion. By default, these protection mechanisms are disabled in BDM; they can be selectively enabled by the FRZ[1:0] bits as shown in **Table 5-3**.

Table 5-3 Effects of FREEZE Assertion

FRZ[1:0]		Disabled Elements
0	0	None
0	1	Bus monitor
1	0	Software watchdog, and periodic interrupt timer
1	1	Both

5.3 System Clock

The system clock in the SCIM2 provides timing signals for IMB modules and the external bus interface. M68HC16 R-series MCUs are fully static MCU designs; register and memory contents are not affected by clock rate changes. System hardware and software support clock rate changes during operation.

5.3.1 System Clock Sources

The system clock signal can be generated from one of three sources. An internal phase-locked loop (PLL) can synthesize the clock from either a slow or fast reference, or the clock signal can be input directly from an external frequency source.

The slow reference is typically a 32.768 kHz crystal; the fast reference is typically a 4.194 MHz crystal. The slow and fast references may be provided by sources other than a crystal. Keep these clock sources in mind while reading the rest of this section.

The system clock source is determined upon $\overline{\text{RESET}}$ assertion by the state of the V_{DDSYN}/MODCLK and FASTREF/PF0 pins. In addition to selecting the system clock source, these pins govern the functionality of the W, X, and Y bits in the synthesizer control register (SYNCR) and the equation that determines the MCU operating frequency. **Table 5-4** summarizes this information.

Table 5-4 System Clock Sources

V_{DDSYN}/ MODCLK	FASTREF/ PF0	Clock Mode	SYNCR W/X/Y Bit Assignments and Reset Values	MCU Operating Frequency Equation
0	X[1]	External Clock	X = %1 Y[2:0] = %000	$f_{sys} = \frac{f_{ref}}{(2-X)(2^{Y})}$, for $Y \leq 6$
1	0	Slow Reference	W = %0 X = %0 Y[5:0] = %111111	$f_{sys} = 4f_{ref}(Y+1)(2^{(2W+X)})$
1	1	Fast Reference	W[2:0] = %011 X = %0 Y[2:0] = %000	$f_{sys} = \frac{f_{ref}(W+1)}{(2-X)(2^{Y})}$, for Y ð 6

NOTES:
1. In external clock mode, the FASTREF/PF0 pin has no effect on clock operation.

The parameter "f_{ref}" refers to the frequency of the clock source connected to the EXTAL pin. The parameter "f_{sys}" refers to the operating frequency of the MCU and has a defined relationship to f_{ref} that depends on the clock mode selected during reset.

The original SCIM does not provide the same flexible system clock generation options as the SCIM2. For example, the phase-locked loop (PLL) on the original SCIM supports only slow reference mode. Operation with a fast reference crystal is supported on some devices by a metal mask option that places a divide-by-128 circuit after the output of the crystal oscillator. A slow reference is thus provided to the PLL, and the same equation, divided by 128, governs the system clock frequency. External clock mode is also supported on the original SCIM, but no additional dividers are present to allow clock speed changes. Consequently, the MCU operates only at the frequency input on the EXTAL pin.

5.3.2 Clock Synthesizer Operation

V_{DDSYN}/MODCLK and V_{SSSYN} are used to power the clock circuits when the system clock is synthesized from a slow or fast reference. Separate clock synthesizer power and ground pins increase MCU noise immunity and allow the clock to run when the MCU is in low-power stop mode. Consequently, V_{DDSYN}/MODCLK should be bypassed to V_{SSSYN} with external capacitors placed as close to the MCU as possible. This provides the clock synthesizer circuits with a quiet power supply and ensures system clock stability.

A crystal oscillator, phase-locked loop (PLL), and frequency dividers outside the PLL make up the clock synthesizer. When power is applied to V_{DDSYN}, a crystal connected to the oscillator's EXTAL and XTAL pins provides a reference frequency to the PLL. The PLL generates the system clock, and the dividers outside the PLL allow changes in clock speed without a PLL relock delay.

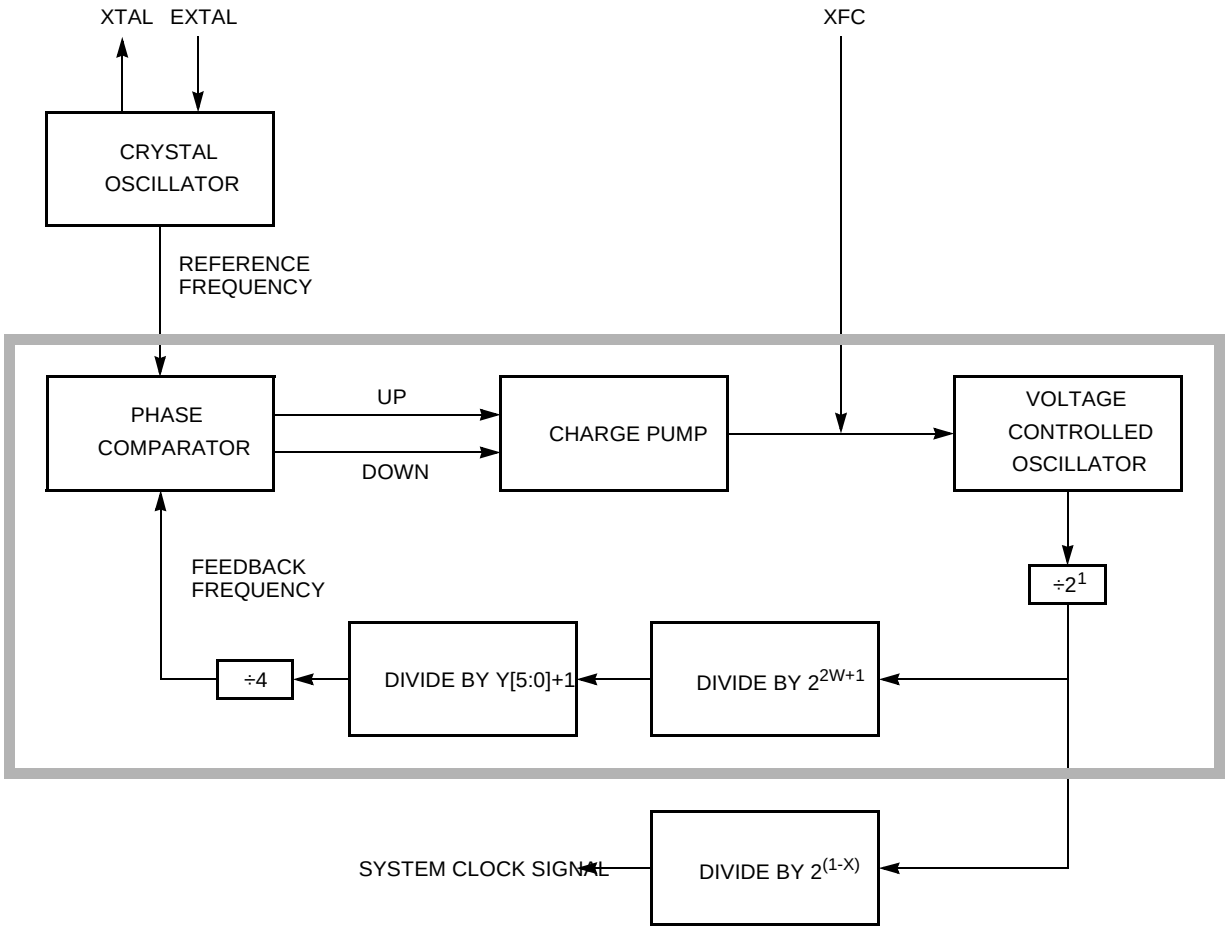
Inside the PLL, a voltage controlled oscillator (VCO) runs at an integral multiple of the MCU operating frequency. The VCO output is fed to two sets of dividers. The first set of dividers is inside the PLL feedback path and serves two purposes. An initial divide-by-two circuit guarantees that the system clock will have a 50% duty cycle. Additional dividers scale the VCO output to provide feedback to the phase comparator. The second set of dividers is outside the PLL and permits MCU operating frequency changes without the need for PLL relock.

The phase comparator produces up and down correction outputs proportional to the phase difference between the reference frequency from the crystal oscillator and the scaled feedback frequency from the VCO. The charge pump integrates and low-pass filters these signals to generate the VCO correction factor. The PLL locks when the divided feedback frequency from the VCO is equal to the reference frequency from the crystal oscillator. The synthesizer lock bit (SLOCK) in SYNCR will be set to one when this occurs.

In slow and fast reference modes, the configuration of dividers inside and outside of the PLL varies. Although the PLL is disabled in external clock mode, the dividers present outside of the PLL in fast reference mode are enabled and can be used to reduce system clock frequency on the fly. Diagrams showing the configuration of the clock synthesizer in each operating mode appear in the sections that follow.

5.3.3 Slow Reference Mode

In slow reference mode, the system clock is generated on-chip by the PLL from an external reference (f_{ref}) that is nominally 32.768 kHz, but can range from 25 kHz to 50 kHz. The frequency of the system clock is controlled by programming the W, X, and Y[5:0] bits in SYNCR. **Figure 5-2** shows the PLL configuration in slow reference mode.



NOTES:
1. THIS DIVIDE BY TWO CIRCUIT GUARANTEES THAT THE SYSTEM CLOCK HAS A DUTY CYCLE OF 50%. IT CANNOT BI

SLOW REF PLL CON

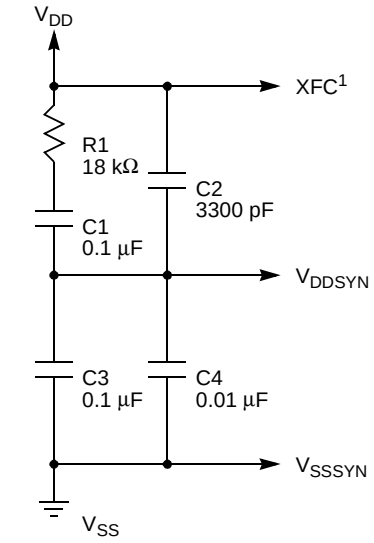
Figure 5-2 Slow Reference Mode PLL Configuration

The Y[5:0] and W bits control dividers inside the PLL that determine the VCO frequency. Changes to these bits require PLL relock. The X bit lies outside the PLL and allows doubling (X = 1) or halving (X = 0) of the system clock frequency without disturbing the VCO. System clock frequency is determined by the following equation:

$$f_{sys} = 4f_{ref}(Y + 1)(2^{(2W + X)})$$

In slow reference mode, the reset state of the upper byte of SYNCR ($3F) results in a power-on f_{sys} of 8.389 MHz when f_{ref} is 32.768 kHz.

In addition to powering V_{DDSYN}/MODCLK with an adequately filtered supply, slow reference mode is selected by driving FASTREF/PF0 to logic zero during reset. PLL lock time and system clock stability are affected by V_{DDSYN}/MODCLK bypassing and design of the XFC low-pass filter. **Figure 5-3** shows one possible bypass/filter circuit for V_{DDSYN}/MODCLK and XFC. Filter circuit implementation can vary, depending on the operating environment, reference multiplication factor, and required clock stability. XFC pin leakage must be kept as low as possible to maintain optimum PLL performance.



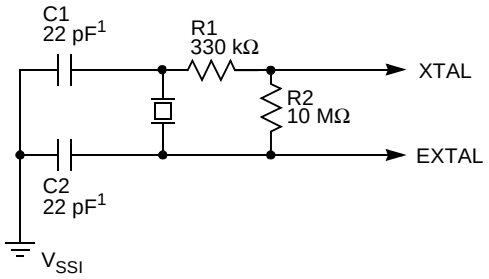
NOTES:
1. MAINTAIN LOW LEAKAGE ON XFC NODE.

SLOW REF BYPASS FIL

Figure 5-3 Slow Reference Mode Bypass/Filter Circuit

The FASTREF/PF0 pin need not be grounded at all times in order for the MCU to operate with a slow reference. Use one of the circuits shown in **Figures 5-21** and **5-22** to drive FASTREF/PF0 to logic zero while $\overline{\text{RESET}}$ is asserted. A pull-down resistor should not be used for this purpose.

To generate the system clock with a slow reference, connect a crystal between the EXTAL and XTAL pins. **Figure 5-4** shows a recommended circuit.



NOTES:
1. RESISTANCE AND CAPACITANCE BASED ON A TEST CIRCUIT CONSTRUCTED WITH A DAISHINKU DMX-38 32.768-kHz SPECIFIC COMPONENTS MUST BE BASED ON CRYSTAL TYPE. CONTACT CRYSTAL VENDOR FOR EXACT CIRCUIT.

SLOW REF OSC

Figure 5-4 Slow Reference Crystal Circuit

As an alternative to using a crystal, a clock signal meeting CMOS voltage levels can serve as a slow reference. Such a frequency source must be connected to EXTAL only; XTAL must be left floating.

5.3.4 Clock Speed Changes in Slow Reference Mode

In slow reference mode, the system clock frequency can be changed on the fly by programming the W, X, and Y[5:0] bits in SYNCR. Care must be taken when changing the system clock speed to observe MCU and VCO frequency limitations. As noted in **5.3.2 Clock Synthesizer Operation**, the VCO operates at an integral multiple of the system clock frequency. Specifically, VCO frequency is governed by the following equations in slow reference mode:

$$f_{VCO} = 4f_{sys} \text{ if } X = 0$$

or

$$f_{VCO} = 2f_{sys} \text{ if } X = 1$$

Failure to observe these equations when changing clock speed can result in operation of the VCO beyond its maximum rating of two times the maximum system clock frequency.

NOTE

This equation applies only in slow reference mode; a different equation is used in fast reference mode, and the VCO is disabled in external clock mode.

In slow reference mode, system clock frequency is governed by the following equation:

$$f_{sys} = 4f_{ref}(Y + 1)(2^{(2W + X)})$$

Table 5-5 shows clock control multipliers for all possible combinations of the SYNCR W, X, and Y[5:0] bits in slow reference mode. To obtain the system clock frequency,

find the Y[5:0] counter modulus in the leftmost column, then multiply f_{ref} by the value in the appropriate prescaler cell.

NOTE

In **Table 5-5**, shaded cells indicate settings of the SYNCR bits that configure the VCO for operation beyond its maximum rated frequency. Although a particular f_{sys} value corresponding to a shaded cell may be a valid MCU operating frequency, the resulting VCO frequency will violate the maximum specification for f_{VCO}. Use of these settings can result in erratic device behavior and must be avoided.

Table 5-5 Slow Reference Mode Clock Control Multipliers

(Shaded cells indicate W, X, and Y[5:0] bit settings that will configure the VCO for operation beyond its maximum rated frequency on this device.)

Modulus	Prescalers			
Y[5:0]	[W:X] = %00	[W:X] = %01	[W:X] = %10	[W:X] = %11
000000	4	8	16	32
000001	8	16	32	64
000010	12	24	48	96
000011	16	32	64	128
000100	20	40	80	160
000101	24	48	96	192
000110	28	56	112	224
000111	32	64	128	256
001000	36	72	144	288
001001	40	80	160	320
001010	44	88	176	352
001011	48	96	192	384
001100	52	104	208	416
001101	56	112	224	448
001110	60	120	240	480
001111	64	128	256	512
010000	68	136	272	544
010001	72	144	288	576
010010	76	152	304	608
010011	80	160	320	640
010100	84	168	336	672
010101	88	176	352	704
010110	92	184	368	736
010111	96	192	384	768
011000	100	200	400	800
011001	104	208	416	832
011010	108	216	432	864
011011	112	224	448	896

Table 5-5 Slow Reference Mode Clock Control Multipliers (Continued)

(Shaded cells indicate W, X, and Y[5:0] bit settings that will configure the VCO for operation beyond its maximum rated frequency on this device.)

Modulus	Prescalers			
Y[5:0]	[W:X] = %00	[W:X] = %01	[W:X] = %10	[W:X] = %11
011100	116	232	464	928
011101	120	240	480	960
011110	124	248	496	992
011111	128	256	512	1024
100000	132	264	528	1056
100001	136	272	544	1088
100010	140	280	560	1120
100011	144	288	576	1152
100100	148	296	592	1184
100101	152	304	608	1216
100110	156	312	624	1248
100111	160	320	640	1280
101000	164	328	656	1312
101001	168	336	672	1344
101010	172	344	688	1376
101011	176	352	704	1408
101100	180	360	720	1440
101101	184	368	736	1472
101110	188	376	752	1504
101111	192	384	768	1536
110000	196	392	784	1568
110001	200	400	800	1600
110010	204	408	816	1632
110011	208	416	832	1664
110100	212	424	848	1696
110101	216	432	864	1728
110110	220	440	880	1760
110111	224	448	896	1792
111000	228	456	912	1824
111001	232	464	928	1856
111010	236	472	944	1888
111011	240	480	960	1920
111100	244	488	976	1952
111101	248	496	992	1984
111110	252	504	1008	2016
111111	256	512	1024	2048

Table 5-6 shows system clock frequencies for all possible combinations of the SYNCR W, X, and Y[5:0] bits in slow reference mode. Bit settings that result in a particular system clock frequency can be easily found using this table.

NOTE

In **Table 5-6**, shaded cells indicate settings of the SYNCR bits that configure the VCO for operation beyond its maximum rated frequency. Although a particular f_{sys} value corresponding to a shaded cell may be a valid MCU operating frequency, the resulting VCO frequency will violate the maximum specification for f_{VCO}. Use of these settings can result in erratic device behavior and must be avoided.

Table 5-6 Slow Reference Mode Clock Frequencies

(Shaded cells indicate W, X, and Y[5:0] bit settings that will configure the VCO for operation beyond its maximum rated frequency on this device.)

Modulus	Prescalers			
Y[5:0]	[W:X] = %00 ($f_{VCO} = 4f_{sys}$)	[W:X] = %01 ($f_{VCO} = 2f_{sys}$)	[W:X] = %10 ($f_{VCO} = 4f_{sys}$)	[W:X] = %11 ($f_{VCO} = 2f_{sys}$)
000000	0.131 MHz	0.262 MHz	0.524 MHz	1.049 MHz
000001	0.262	0.524	1.049	2.097
000010	0.393	0.786	1.573	3.146
000011	0.524	1.049	2.097	4.194
000100	0.655	1.311	2.621	5.243
000101	0.786	1.573	3.146	6.291
000110	0.918	1.835	3.670	7.340
000111	1.049	2.097	4.194	8.389
001000	1.180	2.359	4.719	9.437
001001	1.311	2.621	5.243	10.486
001010	1.442	2.884	5.767	11.534
001011	1.573	3.146	6.291	12.583
001100	1.704	3.408	6.816	13.631
001101	1.835	3.670	7.340	14.680
001110	1.966	3.932	7.864	15.729
001111	2.097	4.194	8.389	16.777
010000	2.228	4.456	8.913	17.826
010001	2.359	4.719	9.437	18.874
010010	2.490	4.981	9.961	19.923
010011	2.621	5.243	10.486	20.972
010100	2.753	5.505	11.010	22.020
010101	2.884	5.767	11.534	23.069
010110	3.015	6.029	12.059	24.117
010111	3.146	6.291	12.583	25.166
011000	3.277	6.554	13.107	26.214
011001	3.408	6.816	13.631	27.263
011010	3.539	7.078	14.156	28.312
011011	3.670	7.340	14.680	29.360
011100	3.801	7.602	15.204	30.409
011101	3.932	7.864	15.729	31.457

Table 5-6 Slow Reference Mode Clock Frequencies (Continued)

(Shaded cells indicate W, X, and Y[5:0] bit settings that will configure the VCO for operation beyond its maximum rated frequency on this device.)

Modulus	Prescalers			
Y[5:0]	[W:X] = %00 ($f_{VCO} = 4f_{sys}$)	[W:X] = %01 ($f_{VCO} = 2f_{sys}$)	[W:X] = %10 ($f_{VCO} = 4f_{sys}$)	[W:X] = %11 ($f_{VCO} = 2f_{sys}$)
011110	4.063	8.126	16.253	32.506
011111	4.194	8.389	16.777	33.554
100000	4.325	8.651	17.302	34.603
100001	4.456	8.913	17.826	35.652
100010	4.588	9.175	18.350	36.700
100011	4.719	9.437	18.874	37.749
100100	4.850	9.699	19.399	38.797
100101	4.981	9.961	19.923	39.846
100110	5.112	10.224	20.447	40.894
100111	5.243	10.486	20.972	41.943
101000	5.374	10.748	21.496	42.992
101001	5.505	11.010	22.020	44.040
101010	5.636	11.272	22.544	45.089
101011	5.767	11.534	23.069	46.137
101100	5.898	11.796	23.593	47.186
101101	6.029	12.059	24.117	48.234
101110	6.160	12.321	24.642	49.283
101111	6.291	12.583	25.166	50.332
110000	6.423	12.845	25.690	51.380
110001	6.554	13.107	26.214	52.429
110010	6.685	13.369	26.739	53.477
110011	6.816	13.631	27.263	54.526
110100	6.947	13.894	27.787	55.575
110101	7.078	14.156	28.312	56.623
110110	7.209	14.418	28.836	57.672
110111	7.340	14.680	29.360	58.720
111000	7.471	14.942	29.884	59.769
111001	7.602	15.204	30.409	60.817
111010	7.733	15.466	30.933	61.866
111011	7.864	15.729	31.457	62.915
111100	7.995	15.991	31.982	63.963
111101	8.126	16.253	32.506	65.012
111110	8.258	16.515	33.030	66.060
111111	8.389	16.777	33.554	67.109

When changing the W and Y[5:0] bits in slow reference mode to increase the system clock speed, a frequency overshoot of as much as 30% can occur before the PLL re-locks. This can be avoided by using the following procedure:

1. Determine values for W and Y[5:0] that will result in the desired frequency when

the X bit is set.
2. With the X bit *cleared*, write the values for W and Y[5:0] into SYNCR.
3. After the PLL locks, *set* the X bit in SYNCR. This will remove a divide by two circuit from the output of the PLL and set the system clock to the desired frequency.

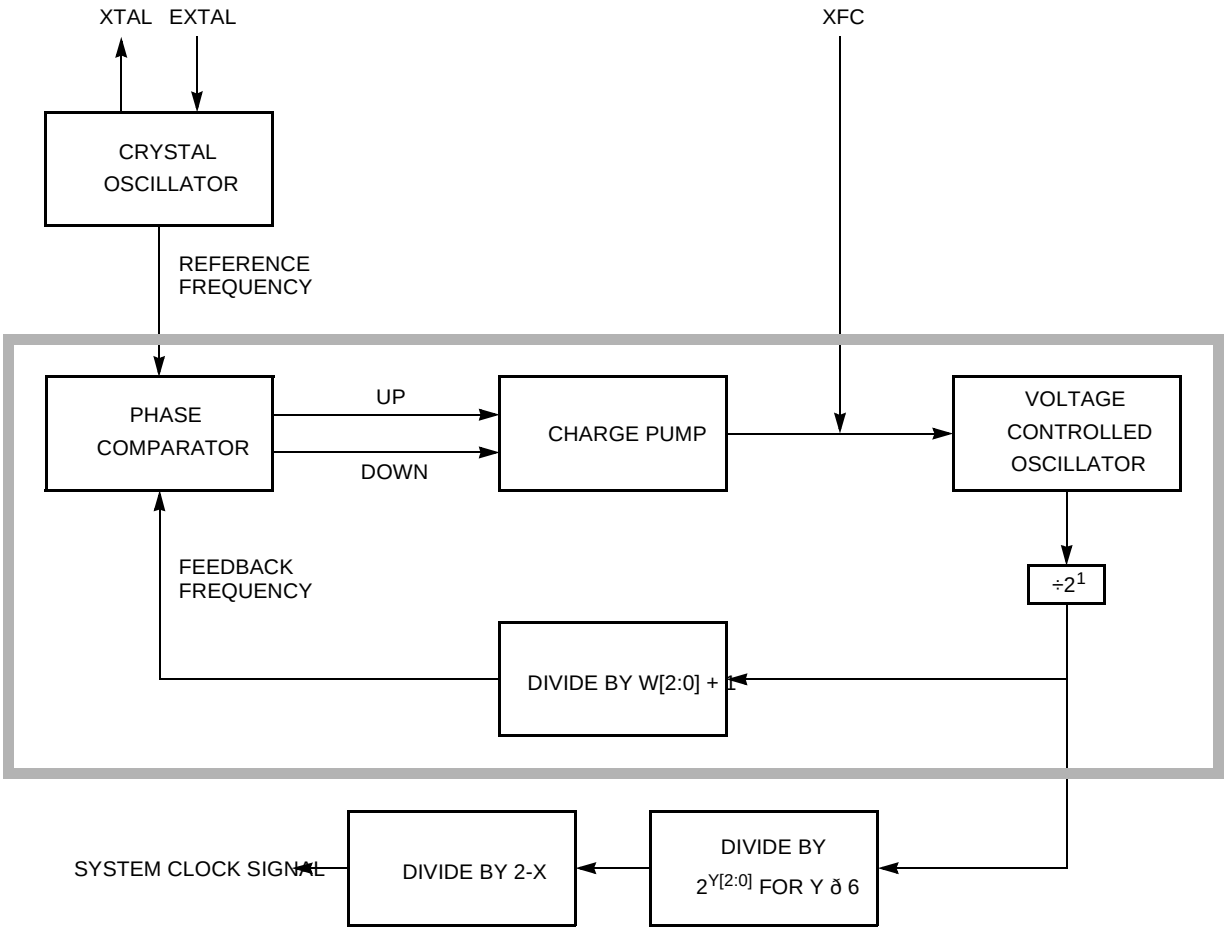
The following example illustrates this in slow reference mode by increasing f_{sys} from its power-on default of 8.389 MHz to 12.059 MHz:

1. From **Table 5-6**, values of W = %0 and Y[5:0] = %101101 will set f_{sys} to 12.059 MHz when X = %1.
2. Write to SYNCR with X = %0, W = %0, and Y[5:0] = %101101.
3. Allow the PLL to re-lock by waiting for the SLOCK bit in SYNCR to set. Now f_{sys} will be 6.029 MHz.
4. Setting the X bit will remove a divide by two circuit from the output of the PLL and double the system clock frequency to 12.059 MHz.

Whether increasing or decreasing the system clock speed, any changes to the W and Y[5:0] bits in slow reference mode also change the VCO frequency. Be sure to allow the PLL to re-lock before proceeding with tasks dependent on stable clock operation.

5.3.5 Fast Reference Mode

In fast reference mode, the system clock is generated on-chip by the PLL from an external reference (f_{ref}) that is nominally 4.194 MHz, but can range from 1 MHz to 6.25 MHz. The frequency of the system clock is controlled by programming the W[2:0], X, and Y[2:0] bits in SYNCR. **Figure 5-5** shows the PLL configuration in fast reference mode.



NOTES:
1. THIS DIVIDE-BY-TWO CIRCUIT GUARANTEES THAT THE SYSTEM CLOCK HAS A DUTY CYCLE OF 50%. IT CANNC

FAST REF PLL CONF

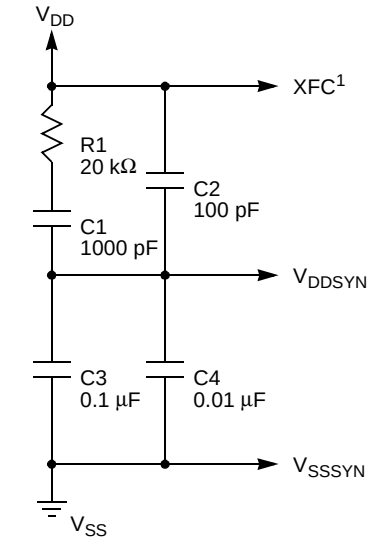
Figure 5-5 Fast Reference Mode PLL Configuration

The W[2:0] bits control a divider inside the PLL that determines the VCO frequency. Changes to these bits require PLL relock. The X and Y[2:0] bits allow doubling (X = 1), halving (X = 0), and division by powers of 2^Y of the system clock frequency. Y can range from 0 to 6; setting Y to 7 has the same effect as setting it to 6. These bits control dividers outside the PLL and can be changed without disturbing the VCO. System clock frequency is determined by the following equation:

$$f_{sys} = \frac{f_{ref}(W + 1)}{(2 - X)(2^Y)}, \; Y \leq 6$$

In fast reference mode, the reset state of the upper byte of SYNCR ($30) results in a power-on f_{sys} of 8.389 MHz when f_{ref} is 4.194 MHz.

In addition to powering V_{DDSYN}/MODCLK with an adequately filtered supply, fast reference mode is selected by driving FASTREF/PF0 to logic one during reset. PLL lock time and system clock stability are affected by V_{DDSYN}/MODCLK bypassing and design of the XFC low-pass filter. **Figure 5-6** shows one possible bypass/filter circuit for V_{DDSYN}/MODCLK and XFC. Filter circuit implementation can vary, depending on the operating environment, reference multiplication factor, and required clock stability. XFC pin leakage must be kept as low as possible to maintain optimum PLL performance.

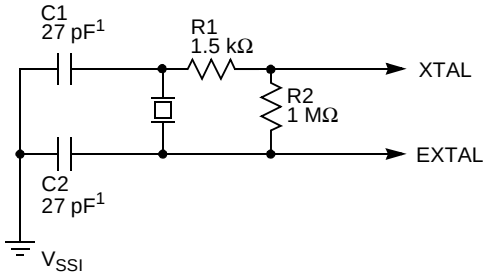


NOTES:
1. MAINTAIN LOW LEAKAGE ON XFC NODE.

FAST REF BYPASS FIL

Figure 5-6 Fast Reference Mode Bypass/Filter Circuit

To generate the system clock with a slow reference, connect a crystal between the EXTAL and XTAL pins. **Figure 5-7** shows a recommended circuit.



NOTES:
1. RESISTANCE AND CAPACITANCE BASED ON A TEST CIRCUIT CONSTRUCTED WITH A KDS041-18 4.194 MHz CRYSTA SPECIFIC COMPONENTS MUST BE BASED ON CRYSTAL TYPE. CONTACT CRYSTAL VENDOR FOR EXACT CIRCUIT.

FAST REF OSC

Figure 5-7 Fast Reference Crystal Circuit

As an alternative to using a crystal, a clock signal meeting CMOS voltage levels can serve as a fast reference. Such a frequency source must be connected to EXTAL only; XTAL must be left floating.

5.3.6 Clock Speed Changes in Fast Reference Mode

In fast reference mode, the system clock frequency can be changed on the fly by programming the W[2:0], X, and Y[2:0] bits in SYNCR. Care must be taken when changing the system clock speed to observe MCU and VCO frequency limitations. As noted in **5.3.2 Clock Synthesizer Operation**, the VCO operates at an integral multiple of the system clock frequency. Specifically, VCO frequency is governed by the following equations in fast reference mode:

$$f_{VCO} = 4(2^{Y})(f_{sys}) = 2^{Y+2}(f_{sys}) \text{ if } X = 0$$

or

$$f_{VCO} = 2(2^{Y})(f_{sys}) = 2^{Y+1}(f_{sys}) \text{ if } X = 1$$

Failure to observe these equations when changing clock speed can result in operation of the VCO beyond its maximum rating of two times the maximum system clock frequency.

NOTE

This equation applies only in fast reference mode; a different equation is used in slow reference mode, and the VCO is disabled in external clock mode.

In fast reference mode, system clock frequency is governed by the following equation:

$$f_{sys} = \frac{f_{ref}(W+1)}{(2-X)(2^{Y})}, \; Y \leq 6$$

Table 5-7 shows clock control multipliers for all possible combinations of the SYNCR W[2:0], X, and Y[2:0] bits in fast reference mode. To obtain the system clock frequency, find the setting of X and Y[2:0] in the prescaler columns, then multiply f_{ref} by the value in the appropriate modulus cell.

NOTE

In **Table 5-7**, shaded cells indicate settings of the SYNCR bits that configure the VCO for operation beyond its maximum rated frequency. Although a particular f_{sys} value corresponding to a shaded cell may be a valid MCU operating frequency, the resulting VCO frequency will violate the maximum specification for f_{VCO}. Use of these settings can result in erratic device behavior and must be avoided.

Table 5-7 Fast Reference Mode Clock Control Multipliers

(Shaded cells indicate W[2:0], X, and Y[2:0] bit settings that will configure the VCO for operation beyond its maximum rated frequency on this device.)

Prescalers		Modulus							
X	Y[2:0][1]	W[2:0] = %000	W[2:0] = %001	W[2:0] = %010	W[2:0] = %011	W[2:0] = %100	W[2:0] = %101	W[2:0] = %110	W[2:0] = %111
0	000	0.5	1	1.5	2	2.5	3	3.5	4
0	001	0.25	0.5	0.75	1	1.25	1.5	1.75	2
0	010	0.125	0.25	0.375	0.5	0.625	0.75	0.875	1
0	011	0.0625	0.125	0.1875	0.25	0.3125	0.375	0.4375	0.5
0	100	0.03125	0.0625	0.09375	0.125	0.15625	0.1875	0.21875	0.25
0	101	0.015625	0.03125	0.046875	0.0625	0.078125	0.09375	0.109375	0.125
0	11X	0.0078125	0.015625	0.0234375	0.03125	0.0390625	0.046875	0.0546875	0.0625
1	000	1	2	3	4	5	6	7	8
1	001	0.5	1	1.5	2	2.5	3	3.5	4
1	010	0.25	0.5	0.75	1	1.25	1.5	1.75	2
1	011	0.125	0.25	0.375	0.5	0.625	0.75	0.875	1
1	100	0.0625	0.125	0.1875	0.25	0.3125	0.375	0.4375	0.5
1	101	0.03125	0.0625	0.09375	0.125	0.15625	0.1875	0.21875	0.25
1	11X	0.015625	0.03125	0.046875	0.0625	0.078125	0.09375	0.109375	0.125

NOTES:
1. Setting Y[2:0] to %111 has the same effect as setting it to %110.

Table 5-8 shows system clock frequencies for all possible combinations of the SYNCR W[2:0], X, and Y[2:0] bits in fast reference mode. Bit settings that result in a particular system clock frequency can be easily found using this table.

NOTE

In **Table 5-8**, shaded cells indicate settings of the SYNCR bits that configure the VCO for operation beyond its maximum rated frequency. Although a particular f_{sys} value corresponding to a shaded cell may be a valid MCU operating frequency, the resulting VCO frequency will violate the maximum specification for f_{VCO}. Use of these settings can result in erratic device behavior and must be avoided.

Table 5-8 Fast Reference Mode Clock Frequencies

(Shaded cells indicate W[2:0], X, and Y[2:0] bit settings that will configure the VCO for operation beyond its maximum rated frequency on this device.)

Prescalers		Modulus							
X	**Y[2:0][1]**	**W[2:0] = %000**	**W[2:0] = %001**	**W[2:0] = %010**	**W[2:0] = %011**	**W[2:0] = %100**	**W[2:0] = %101**	**W[2:0] = %110**	**W[2:0] = %111**
0	000	2.097 MHz	4.194 MHz	6.291 MHz	8.389 MHz	10.486 MHz	12.583 MHz	14.680 MHz	16.777 MHz
0	001	1.049	2.097	3.146	4.194	5.243	6.291	7.340	8.389
0	010	0.524	1.049	1.573	2.097	2.621	3.146	3.670	4.194
0	011	0.262	0.524	0.786	1.049	1.311	1.573	1.835	2.097
0	100	0.131	0.262	0.393	0.524	0.655	0.786	0.918	1.049
0	101	0.066	0.131	0.197	0.262	0.328	0.393	0.459	0.524
0	11X	0.033	0.066	0.098	0.131	0.164	0.197	0.229	0.262
1	000	4.194	8.389	12.583	16.777	20.972	25.166	29.360	33.554
1	001	2.097	4.194	6.291	8.389	10.486	12.583	14.680	16.777
1	010	1.049	2.097	3.146	4.194	5.243	6.291	7.340	8.389
1	011	0.524	1.049	1.573	2.097	2.621	3.146	3.670	4.194
1	100	0.262	0.524	0.786	1.049	1.311	1.573	1.835	2.097
1	101	0.131	0.262	0.393	0.524	0.655	0.786	0.918	1.049
1	11X	0.066	0.131	0.197	0.262	0.328	0.393	0.459	0.524

NOTES:
1. Setting Y[2:0] to %111 has the same effect as setting it to %110.

When changing the W[2:0] bits in fast reference mode to increase the system clock speed, a frequency overshoot of as much as 30% can occur before the PLL re-locks. This can be avoided by using the following procedure:

1. Determine the values of W[2:0], X, and Y[2:0] that provide the desired system clock frequency. Be sure to select a value for W[2:0] that results in the maximum required system clock frequency for the application when the X bit is set and the Y[2:0] bits are cleared.
2. With the X bit *cleared,* write the new W[2:0] setting into SYNCR. If the desired value of Y[2:0] is greater than or equal to the current value of Y[2:0], write it into SYNCR at the same time as X and W[2:0].
3. Allow the PLL to re-lock by waiting for the SLOCK bit in SYNCR to set.
4. Set the X bit to remove a divide by two circuit from the output of the PLL. Also write the new Y[2:0] value into SYNCR at this time if it is less than the previous value.

The following example illustrates this procedure in fast reference mode by increasing f_{sys} from 0.033 MHz with W[2:0] = %000, X = %0, and Y[2:0] = %110 to 16.78 MHz:

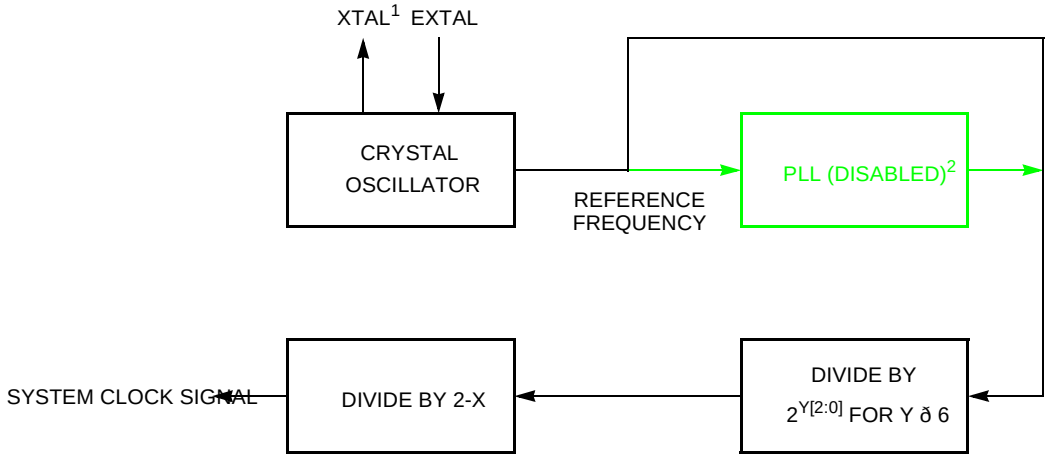
1. From **Table 5-8**, values of W[2:0] = %011, X = %1, and Y[2:0] = %000 will set f_{sys} to 16.78 MHz.
2. With the X bit *cleared,* write W[2:0] = %011 into SYNCR. Do not change Y[2:0] yet.
3. Allow the VCO to re-lock by waiting for the SLOCK bit in SYNCR to set. Now f_{sys} will be 0.131 MHz.

4. Now write to SYNCR with X = %1 and Y[2:0] = %000 to increase the system clock frequency to 16.78 MHz. Although this increases f_{sys} by a factor of 128, PLL re-lock is not needed because the dividers controlled by X and Y[2:0] lie outside of the PLL.

Whether increasing or decreasing the system clock speed, any changes to the W[2:0] in fast reference mode also change the VCO frequency. Be sure to allow the PLL to re-lock before proceeding with tasks dependent on stable clock operation.

5.3.7 External Clock Mode

For applications in which use of the PLL is undesirable, external clock mode allows the system clock signal to be driven onto the EXTAL pin at the same frequency as the maximum required MCU operating frequency. **Figure 5-8** shows the SCIM2 clock configuration in external clock mode.



NOTES:
1. XTAL MUST BE LEFT FLOATING IN EXTERNAL CLOCK MODE.
2. THE PLL IS DISABLED IN EXTERNAL CLOCK MODE BECAUSE VDDSYN/MODCLK IS GROUNDED.

EXT CLK MODE DIV CONF

Figure 5-8 External Clock Mode Divider Configuration

The dividers controlled by the X and Y[2:0] bits in fast reference mode are present in external clock mode and can be used to double (X = 1), halve (X = 0), or divide by powers of 2^Y the input clock frequency (f_{ref}). Y can range from 0 to 6; setting Y to 7 has the same effect as setting it to 6. The upper byte of SYNCR is reset to $80 in external clock mode which results in the power-on f_{sys} being equal to f_{ref}. System clock frequency is determined by the following equation:

$$f_{sys} = \frac{f_{ref}}{(2 - X)(2^Y)}, \; Y \leq 6$$

Operation of the SCIM2 clock circuits is modified in external clock mode. This requires a configuration for the V_{DDSYN}/MODCLK, EXTAL, XTAL, XFC, and FASTREF/PF0 pins that is different from that used in slow and fast reference modes.

First of all, V_{DDSYN}/MODCLK and V_{SSSYN} must be connected to V_{SS}. This places the MCU in external clock mode by disabling the PLL. For this reason, the XFC pin must now be left floating.

Grounding V_{DDSYN}/MODCLK also adjusts the gain of the crystal oscillator so that a CMOS level clock signal must be input on the EXTAL pin. As with the XFC pin, the XTAL pin must be left floating.

Finally, the FASTREF/PF0 pin does not affect MCU configuration and can be connected to other circuitry without special considerations.

In external clock mode, care must be taken to ensure that the signal driven onto the EXTAL pin has a 50% duty cycle, especially when this signal is at or near the maximum MCU operating frequency. The following relationship between the minimum external clock high/low time and period should be observed:

$$\text{Minimum External Clock Period} = \frac{\text{Minimum External Clock High/Low Time}}{50\% - \text{Percentage Variation of External Clock Input Duty Cycle}}$$

5.3.8 M6800 Bus Clock

The state of the E-clock division bit (EDIV) in SYNCR determines the clock rate for the E-clock signal (ECLK) available on pin ADDR23. ECLK is a bus clock for M6800 devices and peripherals. The ECLK frequency can be set to the system clock frequency divided by eight or the system clock frequency divided by sixteen. The clock is enabled by the CS10PA[1:0] field in chip-select pin assignment register 1 (CSPAR1). ECLK operation during low-power stop is described in **5.3.9 Low-Power Operation**.

Refer to **5.9 Chip-Selects** for more information about the M6800 bus clock.

5.3.9 Low-Power Operation

Low-power operation is initiated by the CPU16. To reduce power consumption selectively, the CPU16 can set the STOP bits in each module configuration register. To minimize overall microcontroller power consumption, the CPU16 can execute the LPSTOP instruction which causes the SCIM2 to turn off the system clock.

When the CPU executes LPSTOP, a special CPU space bus cycle writes a copy of the current interrupt mask into the clock control logic. The SCIM2 brings the MCU out of low-power stop mode when one of the following exceptions occur:

- $\overline{\text{RESET}}$
- SCIM2 interrupt of higher priority than the stored interrupt mask

Refer to **5.6.4.2 LPSTOP Broadcast Cycle** for more information.

During low-power stop mode, unless the system clock signal is supplied by an external source and that source is removed, the SCIM2 clock control logic and the SCIM2 clock signal (SCIMCLK) continue to operate. The periodic interrupt timer and input logic for the $\overline{\text{RESET}}$ and $\overline{\text{IRQ}}$ pins are clocked by SCIMCLK. The SCIM2 can also continue to generate the CLKOUT signal while in low-power stop mode.

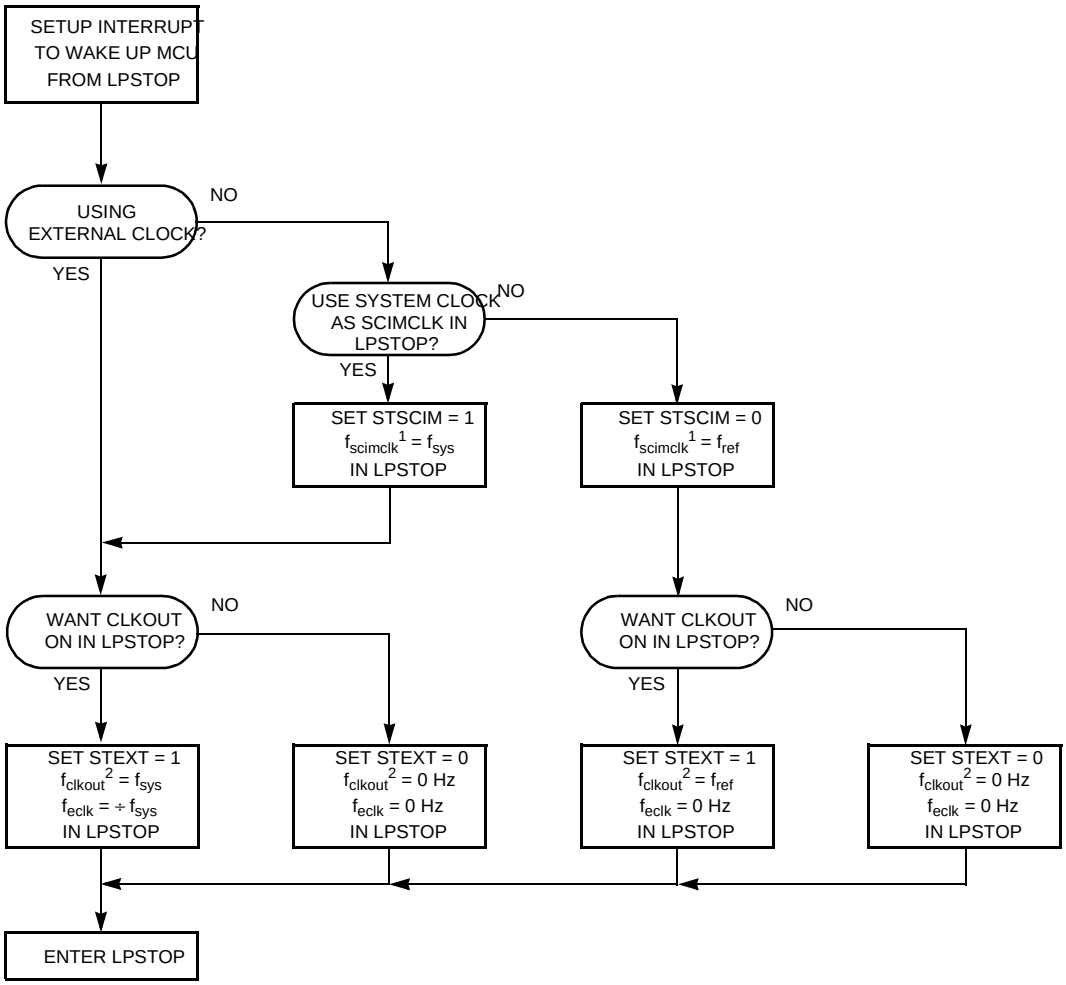
During low-power stop mode, the address bus and data bus continue to drive the LPSTOP broadcast cycle, and bus control signals are negated. I/O pins configured as outputs continue to hold their previous state; I/O pins configured as inputs will remain in a high impedance state.

The STSCIM and STEXT bits in SYNCR determine clock operation during low power stop mode.

The flow chart shown in **Figure 5-9** summarizes the effects of the STSCIM and STEXT bits when the MCU enters low-power stop mode. Any clock in the off state is held low. If the synthesizer VCO is turned off during low-power stop mode, PLL relock delay will occur when the MCU exits LPSTOP mode and the VCO is re-enabled.

NOTE

In LPSTOP mode, the crystal oscillator is not disabled and will continue to run.



NOTES:
1. THE SCIMCLK IS USED BY THE PIT, $\overline{IRQ}$, AND INPUT BLOCKS OF THE SCIM2.
2. CLKOUT CONTROL DURING LPSTOP IS OVERRIDDEN BY THE EXOFF BIT IN SCIMCR. IF EXOFF = PIN IS ALWAYS IN A HIGH IMPEDANCE STATE AND STEXT HAS NO EFFECT IN LPSTOP. IF EXOFF IS CONTROLLED BY STEXT IN LPSTOP.

LPSTOPFLOW

Figure 5-9 LPSTOP Flowchart

5.3.10 Loss of Reference Signal

The SCIM2 includes circuitry to detect a loss of the synthesizer f_{ref} signal and to force reset or to allow continued operation from an alternate clock source. The LOSCD, SLIMP, and RSTEN bits in SYNCR control and report the behavior of the clock synthesizer when f_{ref} loss is detected.

In the SCIM2, f_{ref} is compared to the output of an independent free-running RC oscillator to detect failure. The loss of clock detector should always be triggered when f_{ref} falls below 150 Hz and should never be triggered when f_{ref} is above 20 kHz. This range provides a window of uncertainty sufficiently large enough to compensate for variations in the output of the RC oscillator due to processing and/or operating conditions.

The loss of clock detector can be disabled by writing a one to the loss-of-clock oscillator bit (LOSCD). This disables the free-running RC oscillator and prevents f_{ref} loss from being detected. The reset state of LOSCD is zero which enables the RC oscillator and loss of clock detector.

The reset enable bit (RSTEN) determines how the MCU will process a loss of clock detection. The default state out of reset for RSTEN is zero. This forces the clock synthesizer into the limp mode operating state. In limp mode, the RC oscillator used by the loss of clock detector provides the system clock. Limp mode frequency varies from device to device but does not exceed one half the maximum system clock frequency.

When set to one, RSTEN allows the clock synthesizer to reset the MCU when the loss of clock detector triggers. After powering up from a loss of clock reset, the MCU will set the LOC bit in the reset status register (RSR) and begin operation in limp mode.

The limp status bit (SLIMP) in SYNCR indicates that f_{ref} has failed and that the MCU has entered limp mode. SLIMP will remain set until normal f_{ref} operation is restored.

5.4 System Protection

The system protection block reports reset status information, monitors internal bus activity, and provides periodic interrupt generation. **Figure 5-10** is a block diagram of the submodule.

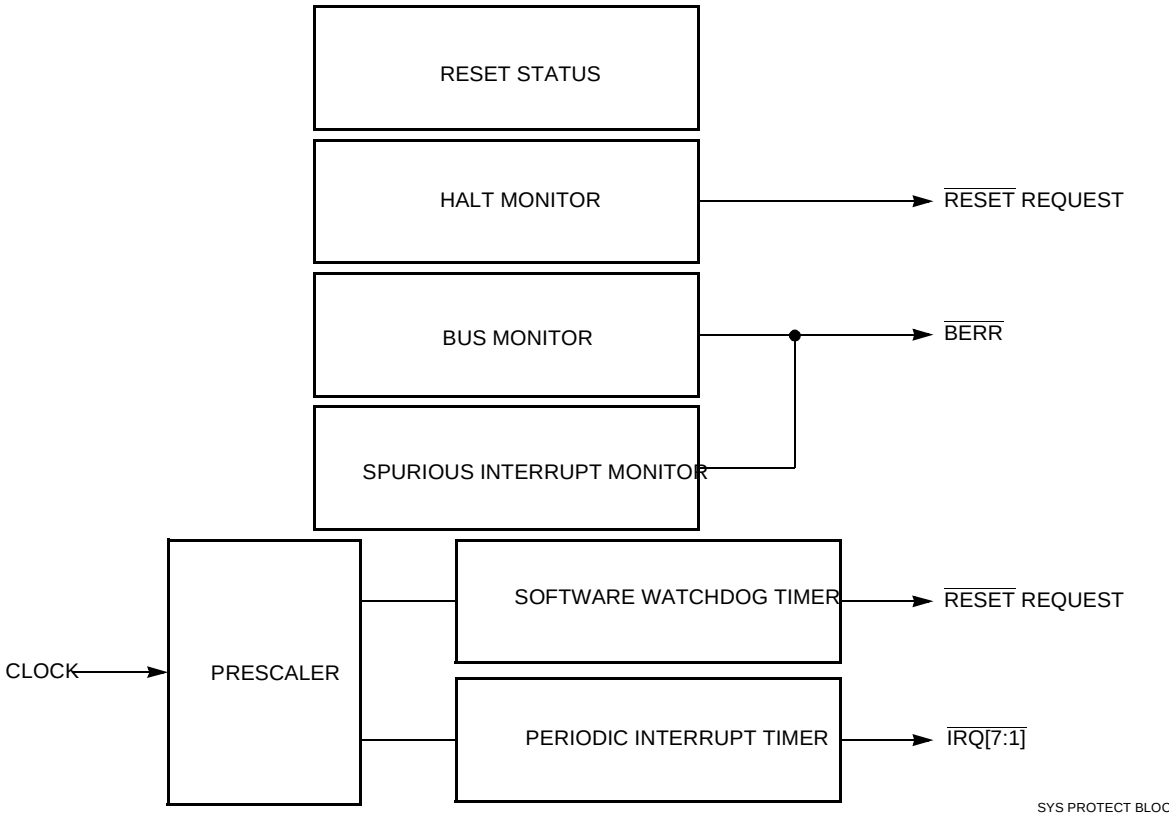


Figure 5-10 System Protection

5.4.1 Reset Status

The reset status register (RSR) latches MCU status during reset. Refer to **5.7.4 Reset Status Register** for more information.

5.4.2 Bus Monitor

The internal bus monitor checks data size acknowledge ($\overline{\text{DSACK}}$) or autovector ($\overline{\text{AVEC}}$) signal response times during normal bus cycles. The monitor asserts the internal bus error ($\overline{\text{BERR}}$) signal when the response time is excessively long.

$\overline{\text{DSACK}}$ and $\overline{\text{AVEC}}$ response times are measured in clock cycles. Maximum allowable response time can be selected by setting the bus monitor timing (BMT[1:0]) field in the system protection control register (SYPCR). **Table 5-9** shows the periods allowed.

Table 5-9 Bus Monitor Period

BMT[1:0]	Bus Monitor Timeout Period
00	64 System clocks
01	32 System clocks
10	16 System clocks
11	8 System clocks

The monitor does not check $\overline{\text{DSACK}}$ response on the external bus unless the CPU16 initiates a bus cycle. The BME bit in SYPCR enables the internal bus monitor for internal to external bus cycles. If a system contains external bus masters, an external bus monitor must be implemented and the internal-to-external bus monitor option must be disabled.

When monitoring transfers to an 8-bit port, the bus monitor does not reset until both byte accesses of a word transfer are completed. Monitor timeout period must be at least twice the number of clocks that a single byte access requires.

5.4.3 Halt Monitor

The halt monitor responds to an assertion of the $\overline{\text{HALT}}$ signal on the internal bus when a double bus fault occurs. A flag in the reset status register (RSR) will indicate when the last reset was caused by the halt monitor. Halt monitor reset can be inhibited by the halt monitor (HME) enable bit in SYPCR. Refer to **5.6.5.2 Double Bus Faults** for more information.

5.4.4 Spurious Interrupt Monitor

During interrupt exception processing, the CPU16 normally acknowledges an interrupt request, arbitrates among various sources of interrupt, recognizes the highest priority source, and then acquires a vector or responds to a request for autovectoring. The spurious interrupt monitor asserts the internal bus error signal ($\overline{\text{BERR}}$) if no interrupt arbitration occurs during interrupt exception processing. The assertion of $\overline{\text{BERR}}$ causes the CPU16 to load the spurious interrupt exception vector into the program counter. The spurious interrupt monitor cannot be disabled.

Refer to **5.8 Interrupts** for further information. For detailed information about interrupt exception processing, refer to **4.13 Exceptions**.

5.4.5 Software Watchdog

The software watchdog is controlled by the software watchdog enable (SWE) bit in SYPCR. When enabled, the watchdog requires that a service sequence be written to the software service register (SWSR) on a periodic basis. If servicing does not take place, the watchdog times out and asserts the $\overline{\text{RESET}}$ signal.

Each time the service sequence is written, the software watchdog timer restarts. The sequence to restart the software watchdog consists of the following steps:

- Write $55 to SWSR.
- Write $AA to SWSR.

Both writes must occur before timeout in the order listed. Any number of instructions can be executed between the two writes.

The clock rate of the watchdog timer is affected by clock mode, the software watchdog prescale (SWP) bit, and the software watchdog timing (SWT[1:0]) field in SYPCR. In slow reference mode and external clock mode, f_{ref} or $f_{ref} \div 512$ can be used to clock the watchdog timer. The options in fast reference mode are $f_{ref} \div 128$ or $(f_{ref} \div 128) \div 512$. In all cases, the divide-by-512 option is selected when SWP = 1.

The value of SWP is affected by the state of the V_{DDSYN}/MODCLK pin during reset, as shown in **Table 5-10**. System software can change SWP value.

Table 5-10 SWP Reset States

V_{DDSYN}/MODCLK	SWP
0 (External Clock)	1 (÷ 512)
1 (Synthesized Clock)	0 (÷ 1)

SWT[1:0] selects the divide ratio used to establish the software watchdog timeout period.

The following equation calculates the timeout period in slow reference mode:

$$\text{Timeout Period} = \frac{\text{Divide Ratio Specified by SWP and SWT[1:0]}}{f_{ref}}$$

The following equation calculates the timeout period in fast reference mode:

$$\text{Timeout Period} = \frac{(128)(\text{Divide Ratio Specified by SWP and SWT[1:0]})}{f_{ref}}$$

The following equation calculates the timeout period in external clock mode:

$$\text{Timeout Period} = \frac{\text{Divide Ratio Specified by SWP and SWT[1:0]}}{f_{ref}}$$

Table 5-11 shows the divide ratio for each combination of the SWP and SWT[1:0] bits. When SWT[1:0] are modified, a watchdog service sequence must be performed before the new timeout period can take effect.

Table 5-11 Software Watchdog Divide Ratio

SWP	SWT[1:0]	Divide Ratio
0	00	2^9
0	01	2^{11}
0	10	2^{13}
0	11	2^{15}
1	00	2^{18}
1	01	2^{20}
1	10	2^{22}
1	11	2^{24}

Figure 5-11 is a block diagram of the watchdog timer and the clock control for the periodic interrupt timer.

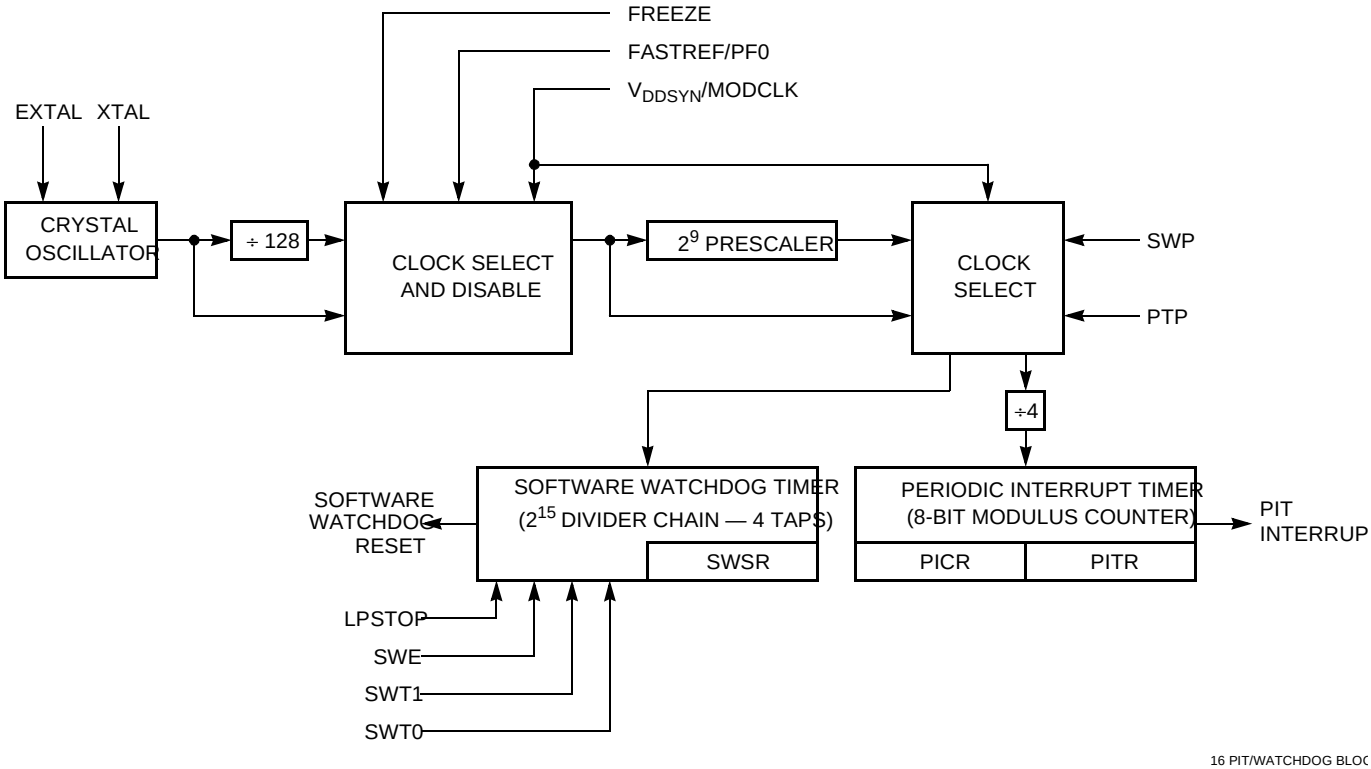


Figure 5-11 Periodic Interrupt Timer and Software Watchdog Timer

5.4.6 Periodic Interrupt Timer

The periodic interrupt timer (PIT) allows the generation of interrupts of specific priority at predetermined intervals. This capability is often used to schedule control system tasks that must be performed within time constraints. The PIT consists of a prescaler, a modulus counter, and registers that determine interrupt timing, priority and vector assignment. Refer to **4.13 Exceptions** for further information about interrupt exception processing.

The periodic interrupt timer modulus counter is clocked by one of two signals. When the PLL is enabled, f_{ref} is used in slow reference mode and $f_{ref} \div 128$ is used in fast reference mode. When the PLL is disabled, f_{ref} is used. The value of the periodic timer prescaler (PTP) bit in the periodic interrupt timer register (PITR) determines system clock prescaling for the periodic interrupt timer. One of two options, either no prescaling, or prescaling by a factor of 512, can be selected. The value of PTP is affected by the state of the V_{DDSYN}/MODCLK pin during reset, as shown in **Table 5-12**. System software can change PTP value.

Table 5-12 PTP Reset States

V_{DDSYN}/MODCLK	PTP
0 (External Clock)	1 (÷ 512)
1 (Synthesized Clock)	0 (÷ 1)

Either clock signal selected by PTP is divided by four before driving the modulus counter. The modulus counter is initialized by writing a value to the periodic interrupt timer modulus (PITM[7:0]) field in PITR. A zero value turns off the PIT. When the modulus counter reaches zero, an interrupt is generated. The modulus counter is then reloaded with the value in PITM[7:0] and counting repeats. If a new value is written to PITR, it is loaded into the modulus counter when the current count is completed.

The following equation calculates the PIT period in slow reference mode:

$$\text{PIT Period} = \frac{(\text{PITM}[7:0])(1 \text{ if PTP} = 0, 512 \text{ if PTP} = 1)(4)}{f_{ref}}$$

The following equation calculates the PIT period in fast reference mode:

$$\text{PIT Period} = \frac{(128)(\text{PITM}[7:0])(1 \text{ if PTP} = 0, 512 \text{ if PTP} = 1)(4)}{f_{ref}}$$

The following equation calculates the PIT period in external clock mode:

$$\text{PIT Period} = \frac{(\text{PITM}[7:0])(1 \text{ if PTP} = 0, 512 \text{ if PTP} = 1)(4)}{f_{ref}}$$

5.4.7 Interrupt Priority and Vectoring for the Periodic Interrupt Timer

Interrupt priority and vectoring for the PIT are determined by the values of the periodic interrupt request level (PIRQL[2:0]) and periodic interrupt vector (PIV[7:0]) fields in the periodic interrupt control register (PICR).

The PIRQL field is compared to the CPU16 interrupt priority mask to determine whether the interrupt is recognized. **Table 5-13** shows PIRQL[2:0] priority values. Because of SCIM2 hardware prioritization, a PIT interrupt is serviced before an external interrupt request of the same priority. The periodic timer continues to run when the interrupt is disabled.

Table 5-13 Periodic Interrupt Priority

PIRQL[2:0]	Priority Level
000	Periodic Interrupt Disabled
001	Interrupt priority level 1
010	Interrupt priority level 2
011	Interrupt priority level 3
100	Interrupt priority level 4
101	Interrupt priority level 5
110	Interrupt priority level 6
111	Interrupt priority level 7

The PIV field contains the periodic interrupt vector. The vector is placed on the IMB when an interrupt request is made. The vector number is used to calculate the address of the appropriate vector in the exception vector table. The reset value of the PIV field is $0F, which corresponds to the uninitialized interrupt exception vector.

5.4.8 Low-Power Stop Mode

When the CPU16 executes the LPSTOP instruction, the current interrupt priority mask is stored in the clock control logic, internal clocks are disabled according to the state of the STSCIM bit in SYNCR, and the MCU enters low-power stop mode. The bus monitor, halt monitor, and spurious interrupt monitor are all inactive during low-power stop.

During low-power stop mode, the clock input to the software watchdog timer is disabled and the timer stops. The software watchdog begins to run again on the first rising clock edge after the MCU exits low-power stop mode. The watchdog is not reset when entering low-power stop mode. A service sequence must be performed to reset the timer.

The periodic interrupt timer does not respond to the LPSTOP instruction, but continues to run during LPSTOP. To stop the periodic interrupt timer, PITM[7:0] must be loaded with zero before entering LPSTOP. A PIT interrupt, or an external interrupt request, can bring the MCU out of low-power stop mode if it has a higher priority than the interrupt mask value stored in the clock control logic when low-power stop mode is entered. LPSTOP can be terminated by a reset.

5.5 External Bus Interface

The external bus interface (EBI) transfers information between the internal MCU bus and external devices. **Figure 5-12** shows a basic system with external memory and peripherals.

The external bus has 24 address lines and 16 data lines. ADDR[19:0] are normal address outputs; ADDR[23:20] follow the output state of ADDR19. The EBI provides dynamic sizing between 8-bit and 16-bit data accesses. It supports byte, word, and long-word transfers. Port width is the maximum number of bits accepted or provided by the external memory system during a bus transfer. Widths of eight and sixteen bits are accessed through the use of asynchronous cycles controlled by the size (SIZ1 and SIZ0) and data size acknowledge ($\overline{\text{DSACK1}}$ and $\overline{\text{DSACK0}}$) pins. Multiple bus cycles may be required for dynamically sized transfers.

To add flexibility and minimize the necessity for external logic, MCU chip-select logic is synchronized with EBI transfers. Refer to **5.9 Chip-Selects** for more information.

5.5.1 Bus Control Signals

The address bus provides addressing information to external devices. The data bus transfers 8-bit and 16-bit data between the MCU and external devices. Strobe signals, one for the address bus and another for the data bus, indicate the validity of an address and provide timing information for data.

Control signals indicate the beginning of each bus cycle, the address space, the size of the transfer, and the type of cycle. External devices can decode these signals and respond to transfer data and terminate the bus cycle. The EBI can operate in an asynchronous mode for any port width.

5.5.1.1 Address Bus

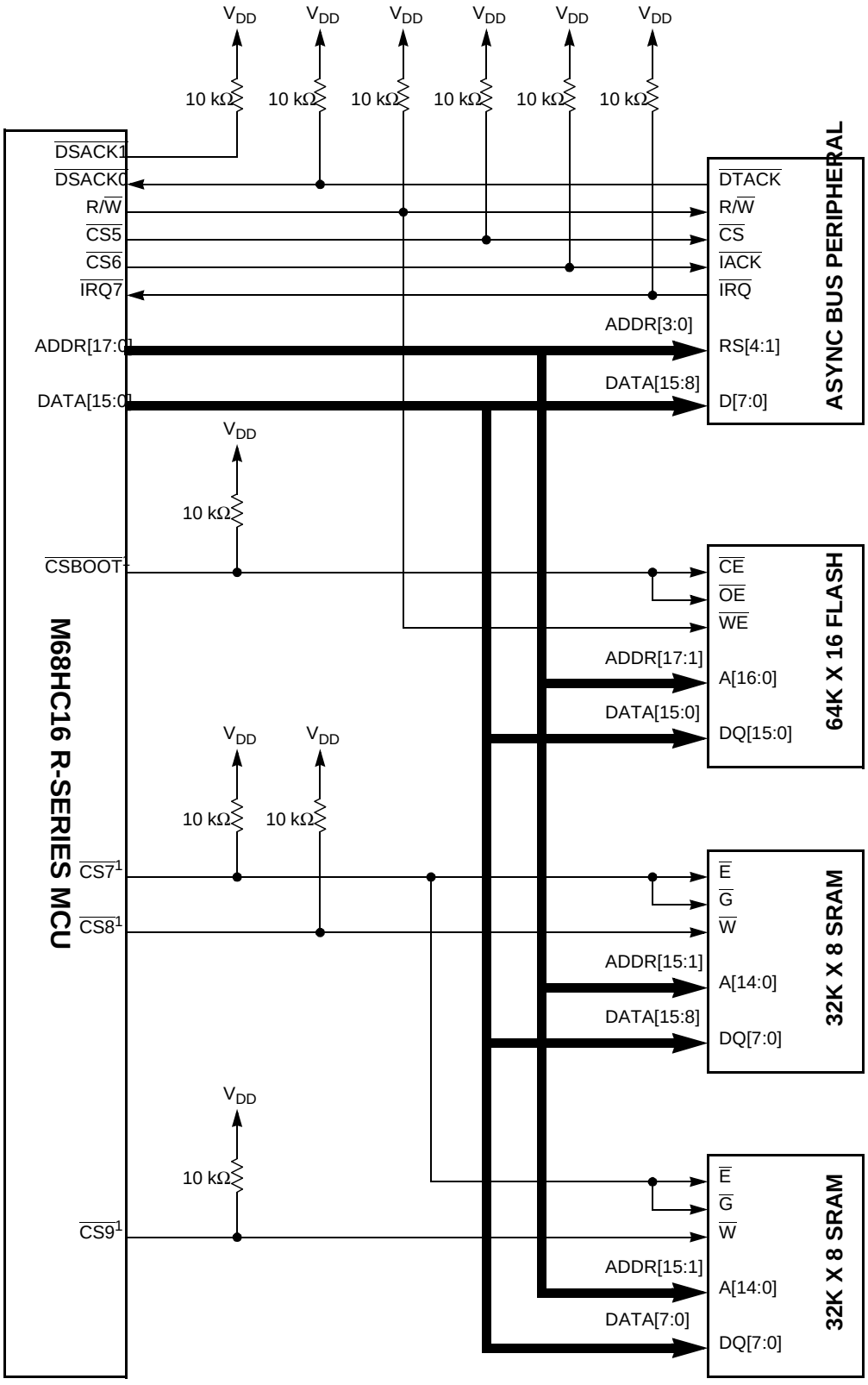
Bus signals ADDR[19:0] define the address of the byte (or the most significant byte) to be transferred during a bus cycle. The MCU places the address on the bus at the beginning of a bus cycle. The address is valid while $\overline{\text{AS}}$ is asserted.

5.5.1.2 Address Strobe

Address strobe ($\overline{\text{AS}}$) is a timing signal that indicates the validity of an address on the address bus and of many control signals.

5.5.1.3 Data Bus

DATA[15:0] form a bidirectional, non-multiplexed parallel bus that transfers data to or from the MCU. A read or write operation can transfer 8 or 16 bits of data in one bus cycle. For a write cycle, all 16 bits of the data bus are driven, regardless of the port width or operand size.



NOTES:
1. THESE CHIP-SELECT LINES ARE CONFIGURED FOR 16-BIT PORT OPERATION IN THIS EXAI

Figure 5-12 MCU Basic System

5.5.1.4 Data Strobe

Data strobe ($\overline{DS}$) is a timing signal. For a read cycle, the MCU asserts $\overline{DS}$ to signal an external device to place data on the bus. $\overline{DS}$ is asserted at the same time as $\overline{AS}$ during a read cycle. For a write cycle, $\overline{DS}$ signals an external device that data on the bus is valid.

5.5.1.5 Read/Write Signal

The read/write signal (R/$\overline{W}$) determines the direction of the transfer during a bus cycle. This signal changes state, when required, at the beginning of a bus cycle, and is valid while $\overline{AS}$ is asserted. R/$\overline{W}$ only transitions when a write cycle is preceded by a read cycle or vice versa. The signal may remain low for two consecutive write cycles.

5.5.1.6 Size Signals

Size signals (SIZ[1:0]) indicate the number of bytes remaining to be transferred during an operand cycle. They are valid while $\overline{AS}$ is asserted. **Table 5-14** shows SIZ0 and SIZ1 encoding.

Table 5-14 Size Signal Encoding

SIZ1	SIZ0	Transfer Size
0	1	Byte
1	0	Word
1	1	3 Byte
0	0	Long word

5.5.1.7 Function Codes

The CPU generates function code signals (FC[2:0]) to indicate the type of activity occurring on the data or address bus. These signals can be considered address extensions that can be externally decoded to determine which of eight external address spaces is accessed during a bus cycle.

Because the CPU16 always operates in supervisor mode (FC2 = 1), address spaces 0 to 3 are not used. Address space 7 is designated CPU space. CPU space is used for control information not normally associated with read or write bus cycles. Function codes are valid while $\overline{AS}$ is asserted. **Table 5-15** shows address space encoding.

Table 5-15 Address Space Encoding

FC2	FC1	FC0	Address Space
1	0	0	Reserved
1	0	1	Data space
1	1	0	Program space
1	1	1	CPU space

5.5.1.8 Data Size Acknowledge Signals

During normal bus transfers, external devices can assert the data size acknowledge signals (DSACK[1:0]) to indicate port width to the MCU. During a read cycle, these signals tell the MCU to terminate the bus cycle and to latch data. During a write cycle, the signals indicate that an external device has successfully stored data and that the cycle can terminate. DSACK[1:0] can also be supplied internally by chip-select logic. Refer to **5.9 Chip-Selects** for more information.

5.5.1.9 Bus Error Signal

The bus error signal (BERR) is asserted when a bus cycle is not properly terminated by DSACK or AVEC assertion. It can also be asserted in conjunction with DSACK to indicate a bus error condition, provided it meets the appropriate timing requirements. Refer to **5.6.5 Alternate Bus Cycle Termination** for more information.

The internal bus monitor can generate the BERR signal for excessively long internal-to-external transfers. In systems with an external bus master, the SCIM2 bus monitor must be disabled and external logic must be provided to drive the BERR pin, because the internal BERR monitor has no information about transfers initiated by an external bus master. Refer to **5.6.6 External Bus Arbitration** for more information.

5.5.1.10 Halt Signal

The halt signal (HALT) can be asserted by an external device for debugging purposes to cause single bus cycle operation. The HALT signal affects external bus cycles only. As a result, a program not requiring use of the external bus may continue executing, unaffected by the HALT signal. When the MCU completes a bus cycle with the HALT signal asserted, DATA[15:0] is placed in a high-impedance state and AS and DS are driven inactive; the address, function code, size, and read/write signals remain in the same state. The MCU does not service interrupt requests while it is halted. Refer to **5.6.5 Alternate Bus Cycle Termination** for further information.

5.5.1.11 Autovector Signal

The autovector signal (AVEC) can be used to terminate external interrupt acknowledgment cycles. Assertion of AVEC causes the CPU16 to generate vector numbers to locate an interrupt handler routine. If AVEC is continuously asserted, autovectors are generated for all external interrupt requests. AVEC is ignored during all other bus cycles. Refer to **5.8 Interrupts** for more information. AVEC for external interrupt requests can also be supplied internally by chip-select logic. Refer to **5.9 Chip-Selects** for more information. The autovector function is disabled when there is an external bus master. Refer to **5.6.6 External Bus Arbitration** for more information.

5.5.2 Dynamic Bus Sizing

The MCU dynamically interprets the port size of an addressed device during each bus cycle, allowing operand transfers to or from 8-bit and 16-bit ports.

During a a bus transfer cycle, an external device signals its port size and indicates completion of the bus cycle to the MCU through the use of the $\overline{\text{DSACK}}$ inputs, as shown in **Table 5-16**. Chip-select logic can generate data size acknowledge signals for an external device. Refer to **5.9 Chip-Selects** for more information.

Table 5-16 Effect of $\overline{\text{DSACK}}$ Signals

$\overline{\text{DSACK1}}$	$\overline{\text{DSACK0}}$	Result
1	1	Insert wait states in current bus cycle
1	0	Complete cycle — Data bus port size is 8 bits
0	1	Complete cycle — Data bus port size is 16 bits
0	0	Reserved

If the CPU is executing an instruction that reads a long-word operand from a 16-bit port, the MCU latches the first 16 bits of valid data and then runs another bus cycle to obtain the other 16 bits. The operation for an 8-bit port is similar, but requires four read cycles. The addressed device uses the $\overline{\text{DSACK}}$ signals to indicate the port width. For instance, a 16-bit external device always returns $\overline{\text{DSACK}}$ for a 16-bit port (regardless of whether the bus cycle is a byte or word operation).

Dynamic bus sizing requires that the portion of the data bus used for a transfer to or from a particular port size be fixed. A 16-bit port must reside on data bus bits [15:0], and an 8-bit port must reside on data bus bits [15:8]. This minimizes the number of bus cycles needed to transfer data and ensures that the MCU transfers valid data.

The MCU always attempts to transfer the maximum amount of data on all bus cycles. For a word operation, it is assumed that the port is 16 bits wide when the bus cycle begins.

Operand bytes are designated as shown in **Figure 5-13**. OP[0:3] represent the order of access. For instance, OP0 is the most significant byte of a long-word operand, and is accessed first, while OP3, the least significant byte, is accessed last. The two bytes of a word-length operand are OP0 (most significant) and OP1. The single byte of a byte-length operand is OP0.

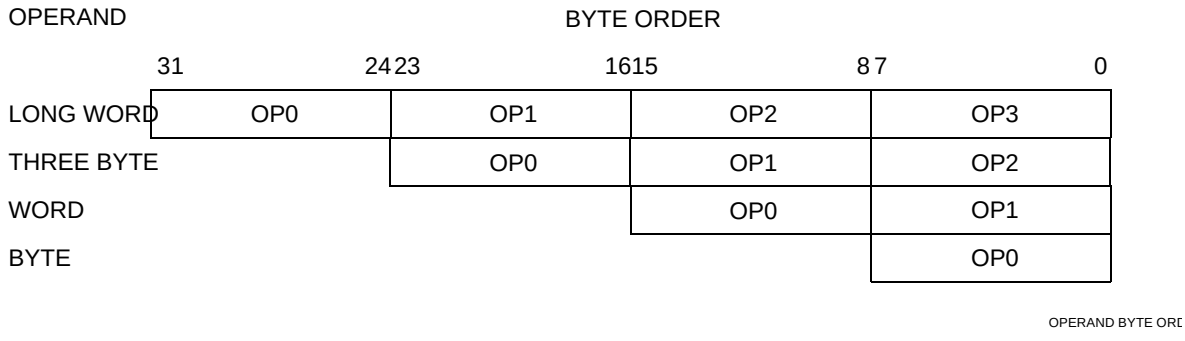


Figure 5-13 Operand Byte Order

5.5.3 Operand Alignment

The EBI data multiplexer establishes the necessary connections for different combinations of address and data sizes. The multiplexer takes the two bytes of the 16-bit bus and routes them to their required positions. Positioning of bytes is determined by the SIZ[1:0] and ADDR0 outputs. SIZ1 and SIZ0 indicate the number of bytes remaining to be transferred during the current bus cycle. The number of bytes transferred is equal to or less than the size indicated by SIZ1 and SIZ0, depending on port width.

ADDR0 also affects the operation of the data multiplexer. During a bus transfer, ADDR[23:1] indicate the word base address of the portion of the operand to be accessed, and ADDR0 indicates the byte offset from the base.

NOTE

ADDR[23:20] follow the state of ADDR19 on CPU16-based MCUs.

5.5.4 Misaligned Operands

The CPU16 uses a basic operand size of 16 bits. An operand is misaligned when it overlaps a word boundary. This is determined by the value of ADDR0. When ADDR0 = 0 (an even address), the address is on a word and byte boundary. When ADDR0 = 1 (an odd address), the address is on a byte boundary only. A byte operand is aligned at any address; a word or long-word operand is misaligned at an odd address.

The largest amount of data that can be transferred by a single bus cycle is an aligned word. If the MCU transfers a long-word operand through a 16-bit port, the most significant operand word is transferred on the first bus cycle and the least significant operand word is transferred on a following bus cycle.

The CPU16 can perform misaligned word transfers. This capability makes it compatible with the M68HC11 CPU. The CPU16 treats misaligned long-word transfers as two misaligned word transfers.

5.5.5 Operand Transfer Cases

Table 5-17 shows how operands are aligned for various types of transfers. OPn entries are portions of a requested operand that are read or written during a bus cycle and are defined by SIZ1, SIZ0, and ADDR0 for that bus cycle. **Table 5-17** also shows to what states $\overline{\text{DSACK}}$[1:0] must be driven — either by a chip select or by external circuitry — to terminate the given bus cycle.

Table 5-17 Operand Alignment

Current Cycle	Transfer Case	SIZ1	SIZ0	ADDR0	$\overline{\text{DSACK1}}$	$\overline{\text{DSACK0}}$	DATA [15:8]	DATA [7:0]	Next Cycle
1	Byte to 8-bit port (even)	0	1	0	1	0	OP0	(OP0)[1]	—
2	Byte to 8-bit port (odd)	0	1	1	1	0	OP0	(OP0)	—
3	Byte to 16-bit port (even)	0	1	0	0	1	OP0	(OP0)	—
4	Byte to 16-bit port (odd)	0	1	1	0	1	(OP0)	OP0	—
5	Word to 8-bit port (aligned)	1	0	0	1	0	OP0	(OP1)	2
6	Word to 8-bit port (misaligned	1	0	1	1	0	OP0	(OP0)	1
7	Word to 16-bit port (aligned)	1	0	0	0	1	OP0	OP1	—
8	Word to 16-bit port (misaligned)	1	0	1	0	1	(OP0)	OP0	3
9	Long word to 8-bit port (aligned)	0	0	0	1	0	OP0	(OP1)	13
10	Long word to 8-bit port (misaligned)[2]	1	0	1	1	0	OP0	(OP0)	1
11	Long word to 16-bit port (aligned)	0	0	0	0	1	OP0	OP1	7
12	Long word to 16-bit port (misaligned)[2]	1	0	1	0	1	(OP0)	OP0	3
13	Three byte to 8-bit port[3]	1	1	1	1	0	OP0	(OP0)	5

NOTES:

1. Operands in parentheses are ignored by the CPU16 during read cycles.
2. The CPU16 treats misaligned long-word transfers as two misaligned-word transfers.
3. Three byte transfer cases occur only as a result of an aligned long word to 8-bit port transfer.

5.6 Bus Operation

Internal microcontroller modules are typically accessed in two system clock cycles. Regular external bus cycles use handshaking between the MCU and external peripherals to manage transfer size and data. These accesses take three system clock cycles, with no wait states. During regular cycles, wait states can be inserted as needed by bus control logic. Refer to **5.6.2 Regular Bus Cycle** for more information.

Fast-termination cycles, which are two clock external accesses with no wait states, use chip-select logic to generate handshaking signals internally. Refer to **5.6.3 Fast Termination Cycles** and **5.9 Chip-Selects** for more information. Bus control signal timing, as well as chip-select signal timing, is specified in **APPENDIX A ELECTRICAL CHARACTERISTICS**. Refer to the *SCIM Reference Manual* (SCIMRM/AD) for more information about each type of bus cycle.

5.6.1 Synchronization to CLKOUT

External devices connected to the MCU bus can operate at a clock frequency different from the frequencies of the MCU as long as the external devices satisfy the interface signal timing constraints. Although bus cycles are classified as asynchronous, they are interpreted relative to the MCU system clock output (CLKOUT).

Descriptions are made in terms of individual system clock states, labelled {S0, S1, S2,..., SN}. The designation “state” refers to the logic level of the clock signal, and does not correspond to any implemented machine state. A clock cycle consists of two successive states. Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for more information.

Bus cycles terminated by $\overline{\text{DSACK}}$ assertion normally require a minimum of three CLKOUT cycles. To support systems that use CLKOUT to generate $\overline{\text{DSACK}}$ and other inputs, asynchronous input setup time and asynchronous input hold times are specified. When these specifications are met, the MCU is guaranteed to recognize the appropriate signal on a specific edge of the CLKOUT signal.

5.6.2 Regular Bus Cycle

The following paragraphs contain a discussion of cycles that use external bus control logic. Refer to **5.6.3 Fast Termination Cycles** for information about fast termination cycles.

To initiate a transfer, the MCU drives the address bus and the SIZ[1:0] signals. The SIZ signals and ADDR0 are externally decoded to select the active portion of the data bus. Refer to **5.5.2 Dynamic Bus Sizing** for more information. When $\overline{\text{AS}}$, $\overline{\text{DS}}$, and R/$\overline{\text{W}}$ are valid, a peripheral device either places data on the bus (read cycle) or latches data from the bus (write cycle), then asserts a $\overline{\text{DSACK}}$[1:0] combination to indicate the port size.

The $\overline{\text{DSACK}}$[1:0] signals can be asserted before the data from a peripheral device is valid on a read cycle. To ensure valid data is latched by the MCU, a maximum period between MCU assertion of $\overline{\text{DS}}$ and user-supplied assertion of $\overline{\text{DSACK}}$[1:0] is specified.

There is no specified maximum for the period between MCU assertion of $\overline{\text{AS}}$ and user-supplied assertion of $\overline{\text{DSACK}}$[1:0]. Although the MCU can transfer data in a minimum of three clock cycles when the cycle is terminated with $\overline{\text{DSACK}}$, the MCU inserts wait cycles in clock period increments until either $\overline{\text{DSACK1}}$ or $\overline{\text{DSACK0}}$ goes low.

If the $\overline{\text{DSACK}}$ bus termination signals remain unasserted, the MCU will continue to insert wait states, and the bus cycle will never end. If no peripheral responds to an access, or if an access is invalid, external logic should assert the $\overline{\text{BERR}}$ or $\overline{\text{HALT}}$ signals to abort the bus cycle (when $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$ are asserted simultaneously, the CPU16 acts as though only $\overline{\text{BERR}}$ is asserted). When enabled, the SCIM2 bus monitor asserts $\overline{\text{BERR}}$ when $\overline{\text{DSACK}}$ response time exceeds a predetermined limit. The bus monitor timeout period is determined by the BMT[1:0] field in SYPCR. The maximum bus monitor timeout period is 64 system clock cycles.

5.6.2.1 Read Cycle

During a read cycle, the MCU transfers data from an external memory or peripheral device. If the instruction specifies a long-word or word operation, the MCU attempts to read two bytes at once. For a byte operation, the MCU reads one byte. The portion of the data bus from which each byte is read depends on operand size, peripheral address, and peripheral port size.

Figure 5-14 is a flow chart of a word read cycle. Refer to **5.5.2 Dynamic Bus Sizing**, **5.5.4 Misaligned Operands**, and the *SCIM Reference Manual* (SCIMRM/AD) for more information.

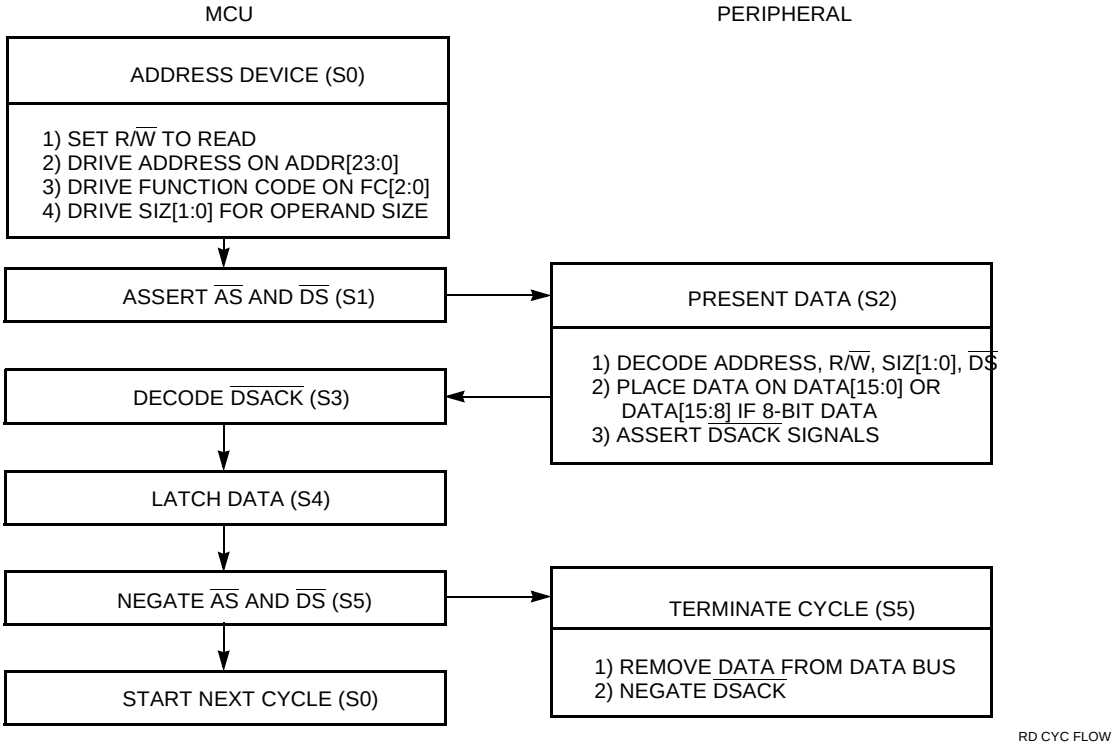

MCU
PERIPHERAL
ADDRESS DEVICE (S0)
1) SET R/W TO READ
2) DRIVE ADDRESS ON ADDR[23:0]
3) DRIVE FUNCTION CODE ON FC[2:0]
4) DRIVE SIZ[1:0] FOR OPERAND SIZE
ASSERT AS AND DS (S1)
PRESENT DATA (S2)
1) DECODE ADDRESS, R/W, SIZ[1:0], DS
2) PLACE DATA ON DATA[15:0] OR DATA[15:8] IF 8-BIT DATA
3) ASSERT DSACK SIGNALS
DECODE DSACK (S3)
LATCH DATA (S4)
NEGATE AS AND DS (S5)
TERMINATE CYCLE (S5)
1) REMOVE DATA FROM DATA BUS
2) NEGATE DSACK
START NEXT CYCLE (S0)
RD CYC FLOW

Figure 5-14 Word Read Cycle Flowchart

5.6.2.2 Write Cycle

During a write cycle, the MCU transfers data to an external memory or peripheral device. If the instruction specifies a long-word or word operation, the MCU attempts to write two bytes at once. For a byte operation, the MCU writes one byte. The portion of the data bus upon which each byte is written depends on operand size, peripheral address, and peripheral port size.

Figure 5-15 is a flow chart of a write-cycle. Refer to **5.5.2 Dynamic Bus Sizing**, **5.5.4 Misaligned Operands**, and the *SCIM Reference Manual* (SCIMRM/AD) for more information.

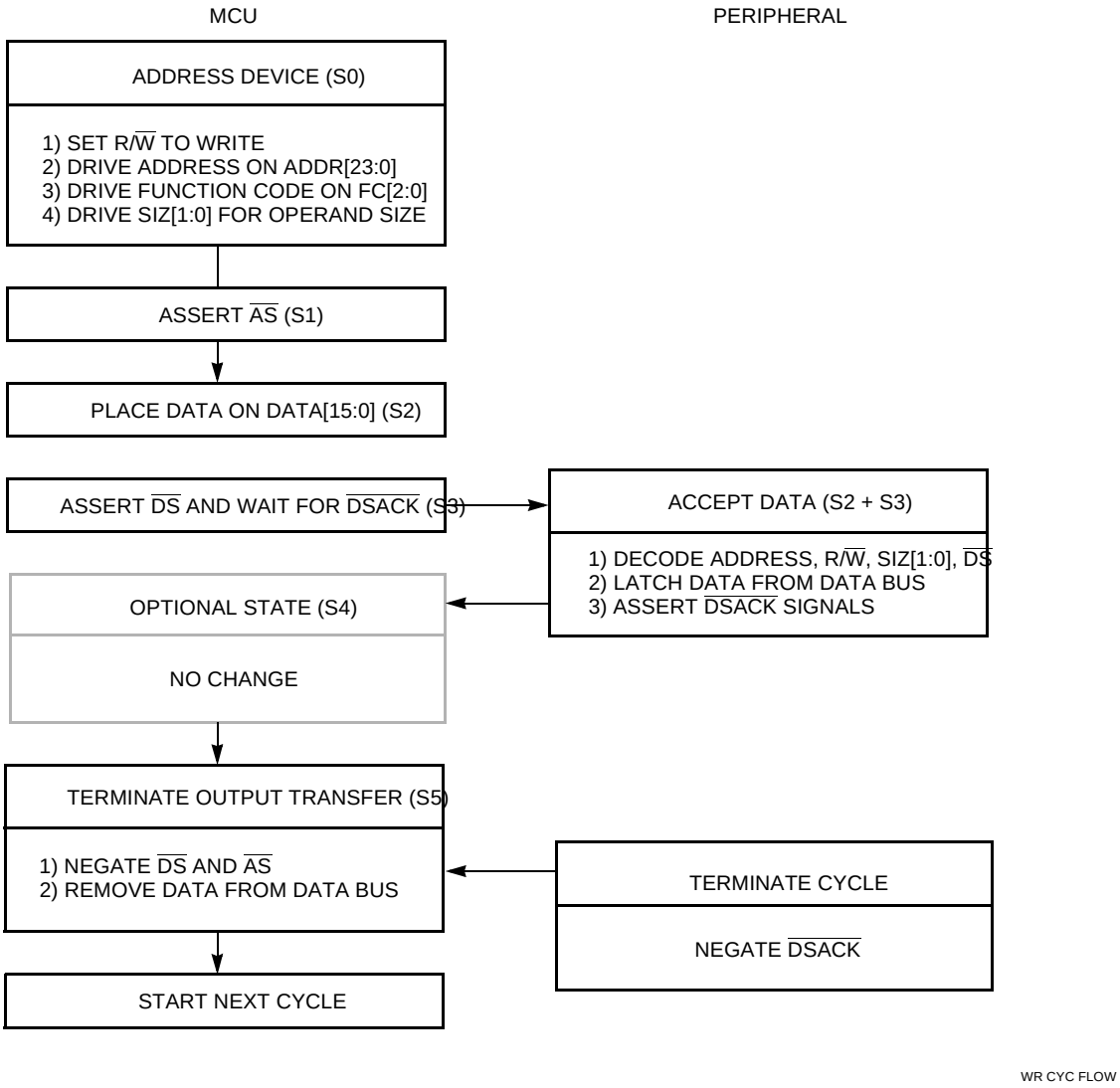

MCU
PERIPHERAL
ADDRESS DEVICE (S0)
1) SET R/W TO WRITE
2) DRIVE ADDRESS ON ADDR[23:0]
3) DRIVE FUNCTION CODE ON FC[2:0]
4) DRIVE SIZ[1:0] FOR OPERAND SIZE
ASSERT AS (S1)
PLACE DATA ON DATA[15:0] (S2)
ASSERT DS AND WAIT FOR DSACK (S3)
ACCEPT DATA (S2 + S3)
1) DECODE ADDRESS, R/W, SIZ[1:0], DS
2) LATCH DATA FROM DATA BUS
3) ASSERT DSACK SIGNALS
OPTIONAL STATE (S4)
NO CHANGE
TERMINATE OUTPUT TRANSFER (S5)
1) NEGATE DS AND AS
2) REMOVE DATA FROM DATA BUS
TERMINATE CYCLE
NEGATE DSACK
START NEXT CYCLE
WR CYC FLOW

Figure 5-15 Write Cycle Flowchart

5.6.3 Fast Termination Cycles

When an external device can meet fast access timing, the fast termination option of SCIM2 chip selects can provide a two cycle external bus transfer. Because the chip-select circuits are driven from the system clock, bus cycle termination is inherently synchronized with the system clock.

If multiple chip-selects are to be used to provide control signals to a single device and match conditions can occur simultaneously, all MODE, STRB, and associated DSACK fields must be programmed to the same value. This prevents a conflict on the internal bus when the wait states are loaded into the DSACK counter shared by all chip-selects.

Fast termination cycles use internal handshaking signals generated by the chip-select logic. To initiate a transfer, the MCU drives the address bus and the SIZ[1:0] signals. When $\overline{AS}$, $\overline{DS}$, and R/$\overline{W}$ are valid, a peripheral device either places data on the bus (read cycle) or latches data from the bus (write cycle). At the appropriate time, chip-select logic asserts the $\overline{DSACK}$[1:0] signals.

The $\overline{DSACK}$ field in the chip-select option registers determine whether internally generated $\overline{DSACK}$ or externally generated $\overline{DSACK}$ is used. For fast termination cycles, the fast termination encoding (%1110) must be used. Refer to **5.9.1 Chip-Select Registers** for information about fast termination setup.

The external $\overline{DSACK}$ lines are always active, regardless of the setting of the $\overline{DSACK}$ field in the chip-select option registers. Thus, an external $\overline{DSACK}$ can always terminate a bus cycle. Holding a $\overline{DSACK}$ line low will cause essentially all external bus cycles to be three-cycle (zero wait states) accesses unless the chip-select option register specifies fast termination accesses.

To use fast termination, an external device must be fast enough to have data ready within the specified setup time (for example, by the falling edge of S4). Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for information about fast termination timing.

When a fast termination cycle is issued, $\overline{DS}$ is asserted for reads but not for writes. The STRB field in the chip-select option register used must be programmed with the address strobe encoding to assert the chip-select signal for a fast termination write.

5.6.4 CPU Space Cycles

Function code signals FC[2:0] designate which of eight external address spaces is accessed during a bus cycle. Address space 7 is designated CPU space. CPU space is used for control information not normally associated with read or write bus cycles. Function codes are valid only while $\overline{AS}$ is asserted. Refer to **5.5.1.7 Function Codes** for more information on codes and encoding.

During a CPU space access, ADDR[19:16] are encoded to reflect the type of access being made. Three encodings are used by the MCU, as shown in **Figure 5-16**. These encodings represent breakpoint acknowledge (Type $0) cycles, low power stop broadcast (Type $3) cycles, and interrupt acknowledge (Type $F) cycles. Type $0 and type $3 cycles are discussed in the following paragraphs. Refer to **5.8 Interrupts** for information about interrupt acknowledge bus cycles.

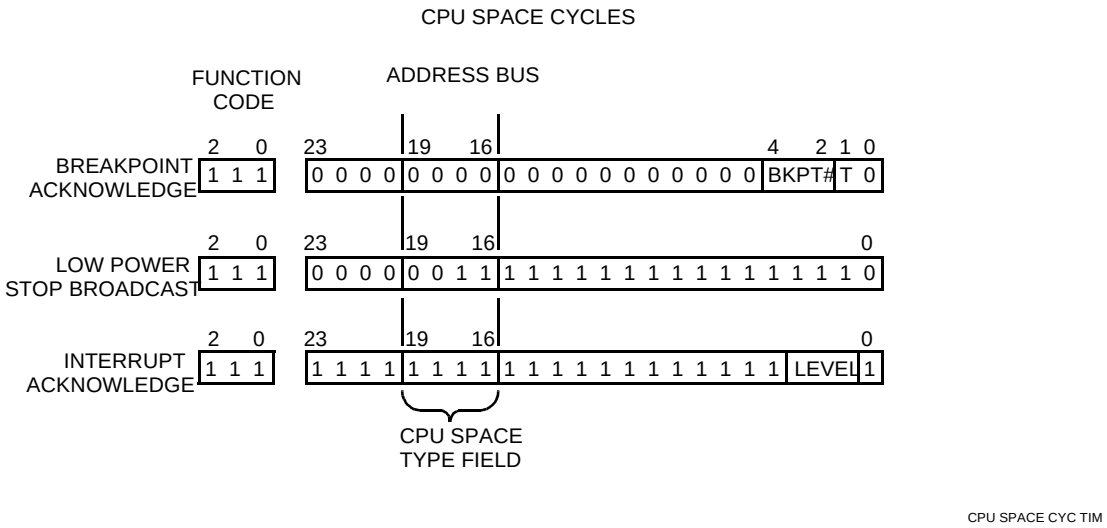


Figure 5-16 CPU Space Address Encoding

5.6.4.1 Breakpoint Acknowledge Cycle

Breakpoints stop program execution at a predefined point during system development. On M68HC16 R-series MCUs, breakpoints are treated as a type of exception. Breakpoints can be used alone or in conjunction with background debug mode.

M68HC16 R-series MCUs have only one source and type of breakpoint. This is a hardware breakpoint initiated by asserting the $\overline{\text{BKPT}}$ input. (CPU32-based microcontrollers also support software breakpoints). The breakpoint acknowledge cycle discussed here is the bus cycle that occurs as a part of breakpoint exception processing when a breakpoint is initiated while background debug mode is not enabled.

$\overline{\text{BKPT}}$ is sampled on the same clock phase as data. If $\overline{\text{BKPT}}$ is valid, the data is tagged as it enters the CPU16 pipeline. When $\overline{\text{BKPT}}$ is asserted while data is valid during an instruction prefetch, the acknowledge cycle occurs immediately after that instruction has executed. When $\overline{\text{BKPT}}$ is asserted while data is valid during an operand fetch, the acknowledge cycle occurs immediately after execution of the instruction during which it is latched. If $\overline{\text{BKPT}}$ is asserted for only one bus cycle and a pipe flush occurs before $\overline{\text{BKPT}}$ is detected by the CPU16, no acknowledge cycle occurs. To ensure detection, $\overline{\text{BKPT}}$ should be asserted until a breakpoint acknowledge cycle is recognized.

The CPU16 responds to $\overline{\text{BKPT}}$ assertion by initiating a breakpoint acknowledge read cycle. SIZ[1:0] and FC[2:0] will be driven to %10 and %111, respectively, indicating a word access to CPU space. The address bus will be driven to $00001E, where:

- ADDR[19:16] = %0000 denotes a breakpoint acknowledge cycle,
- ADDR[4:2] = %111 denotes the breakpoint number (all hardware breakpoints uses number %111), and
- ADDR1 = %1 denotes a hardware breakpoint.

External breakpoint circuitry or a chip select should decode the function code and address lines and assert DSACK[1:0] or BERR to terminate the bus cycle. Data acquired from this read cycle is ignored.

When the breakpoint acknowledge bus cycle is terminated, the CPU16 will begin hardware breakpoint exception processing by:

- Internally acquiring the number of the hardware breakpoint exception vector
- Computing the vector address from the vector number
- Loading the content of the vector address into the program counter
- Jumping to the address of the exception handler

Refer to **Figure 5-17** for a flow chart of hardware breakpoint operation. Refer to the *SCIM Reference Manual* (SCIMRM/AD) for more information.

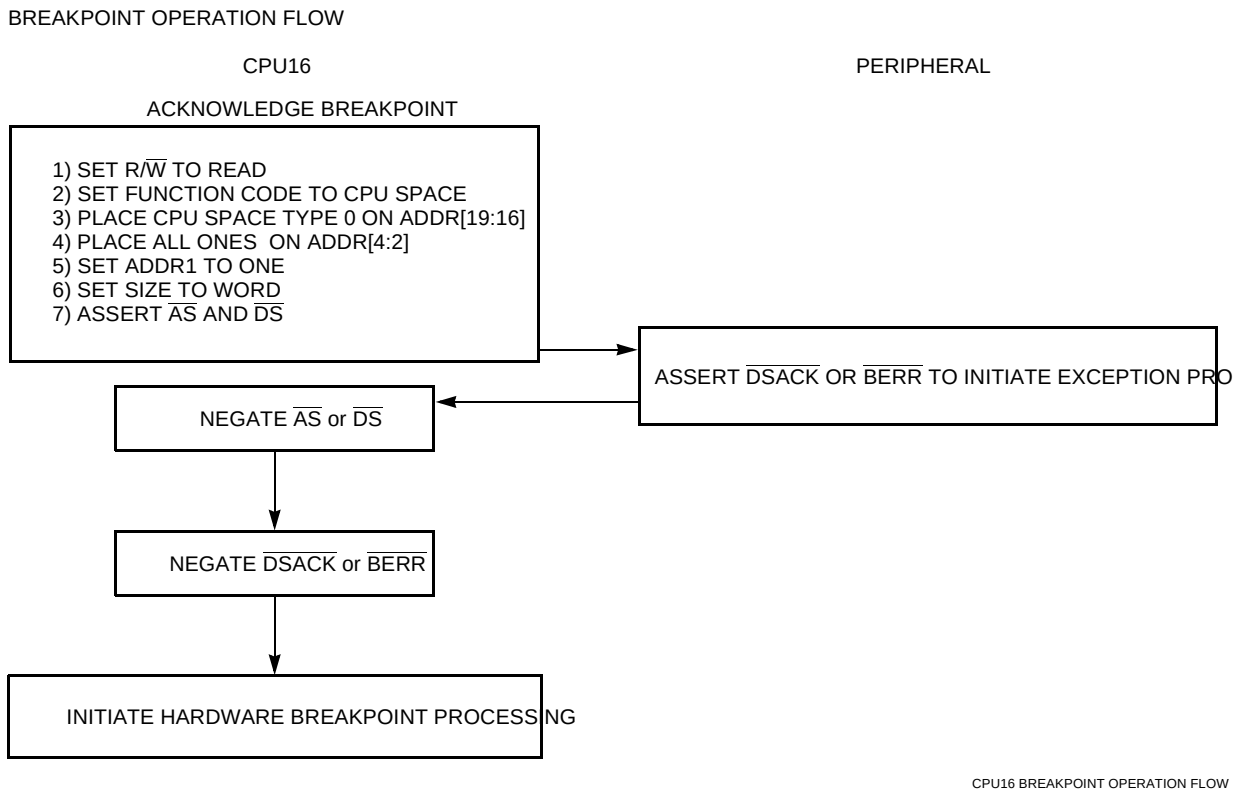

BREAKPOINT OPERATION FLOW
CPU16
PERIPHERAL
ACKNOWLEDGE BREAKPOINT
1) SET R/W TO READ
2) SET FUNCTION CODE TO CPU SPACE
3) PLACE CPU SPACE TYPE 0 ON ADDR[19:16]
4) PLACE ALL ONES ON ADDR[4:2]
5) SET ADDR1 TO ONE
6) SET SIZE TO WORD
7) ASSERT AS AND DS
ASSERT DSACK OR BERR TO INITIATE EXCEPTION PRO
NEGATE AS or DS
NEGATE DSACK or BERR
INITIATE HARDWARE BREAKPOINT PROCESSING
CPU16 BREAKPOINT OPERATION FLOW

Figure 5-17 Breakpoint Operation Flowchart

5.6.4.2 LPSTOP Broadcast Cycle

Low-power stop mode is initiated by the CPU16. Individual modules can be stopped by setting the STOP bits in each module configuration register. The SCIM2 can turn off system clocks after execution of the LPSTOP instruction. When the CPU16 executes LPSTOP, a low-power stop broadcast cycle is generated. The SCIM2 brings the MCU out of low-power mode when either an interrupt of higher priority than the interrupt mask level in the CPU16 condition code register or a reset occurs.

Refer to **5.3.9 Low-Power Operation** and **SECTION 4 CENTRAL PROCESSOR UNIT** for more information.

During an LPSTOP broadcast cycle, the CPU16 performs a CPU space write to address $3FFFE. This write puts a copy of the interrupt mask value in the clock control logic. The mask is encoded on the data bus as shown in **Figure 5-18**.

The LPSTOP CPU space cycle is shown externally (if the bus is available) as an indication to external devices that the MCU is going into low-power stop mode. The SCIM2 provides an internally generated DSACK response to this cycle. The timing of this bus cycle is the same as for a fast termination write cycle. If the bus is not available (arbitrated away), the LPSTOP broadcast cycle is not shown externally.

NOTE

BERR assertion during the LPSTOP broadcast cycle is ignored.

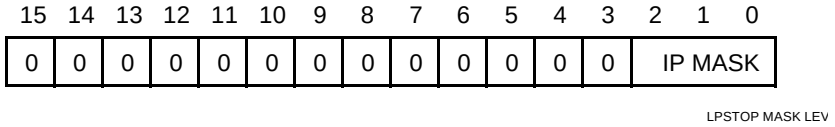


Figure 5-18 LPSTOP Interrupt Mask Encoding on DATA[15:0]

5.6.5 Alternate Bus Cycle Termination

External circuitry or a chip select must assert at least one of the DSACK[1:0] signals or the AVEC signal to terminate a bus cycle normally. Alternate termination methods are used when the bus cycle cannot or must not be terminated in the expected manner. For example, the SCIM2 bus monitor can terminate excessively long cycles by internally asserting the BERR signal. External circuitry can also assert BERR or HALT to force bus cycle termination and interrupt the normal flow of instruction processing. Assertion of BERR or HALT can occur under the following conditions:

- Bus error signal (BERR)
  - — When assertion of DSACK[1:0] or AVEC does not occur within a specified period after assertion of AS, the SCIM2 bus monitor (if enabled) can assert BERR internally.
  - — The spurious interrupt monitor asserts BERR internally when an interrupt request is acknowledged and no interrupt arbitration (IARB) contention occurs. BERR assertion terminates the interrupt acknowledge (IACK) cycle and causes the MCU to generate the spurious interrupt vector number.
  - — External devices can assert BERR to indicate an external bus error.

- Halt signal ($\overline{HALT}$)
  - — $\overline{HALT}$ can be asserted by an external device to cause single bus cycle operation. $\overline{HALT}$ is typically used for debugging purposes.

Table 5-18 summarizes acceptable bus cycle terminations for asynchronous cycles in relation to $\overline{DSACK}$[1:0] assertion.

Table 5-18 $\overline{DSACK}$[1:0], $\overline{BERR}$, and $\overline{HALT}$ Assertion Results

Type of Termination	Control Signal	Asserted on Rising Edge of State		Result
		S[1]	S + 2	
NORMAL	$\overline{DSACK}$[1:0] $\overline{BERR}$ $\overline{HALT}$	A[2] NA[3] NA	RA[4] NA X[5]	Terminate cycle normally and continue.
HALT	$\overline{DSACK}$[1:0] $\overline{BERR}$ $\overline{HALT}$	A NA A/RA	RA NA RA	Terminate cycle normally and halt. Continue when $\overline{HALT}$ is negated.
BUS ERROR	$\overline{DSACK}$[1:0] $\overline{BERR}$ $\overline{HALT}$	X A/NA X	X A/RA X	Terminate cycle and process bus error exception.

NOTES:
1. S = The number of the current even-numbered bus state (e.g. S2, S4, etc.)
2. A = Signal is asserted in this bus state.
3. NA = Signal is not asserted in this bus state.
4. RA = Signal was asserted in previous state and remains asserted in this state.
5. X = Don't care.

5.6.5.1 Bus Errors

The CPU16 treats bus errors as a type of synchronous exception. Bus error exception processing begins when the CPU16 detects assertion of the IMB $\overline{BERR}$ signal.

$\overline{BERR}$ assertions do not force immediate exception processing. The signal is synchronized with normal bus cycles and is latched into the CPU16 at the end of the bus cycle in which it was asserted. Because bus cycles can overlap instruction boundaries, bus error exception processing may not occur at the end of the instruction in which the bus cycle begins. Timing of $\overline{BERR}$ detection/acknowledge is dependent upon several factors:

- Which bus cycle of an instruction is terminated by assertion of $\overline{BERR}$.
- The number of bus cycles in the instruction during which $\overline{BERR}$ is asserted.
- The number of bus cycles in the instruction following the instruction in which $\overline{BERR}$ is asserted.
- Whether $\overline{BERR}$ is asserted during a program space access or a data space access.

Because of these factors, it is impossible to predict precisely how long after occurrence of a bus error the bus error exception is processed.

NOTE

The external bus interface does not latch data when an external bus cycle is terminated by $\overline{\text{BERR}}$. When this occurs during an instruction prefetch, the IMB precharge state (bus pulled high, or $FFFF) is latched into the CPU16 instruction register, with indeterminate results.

5.6.5.2 Double Bus Faults

Exception processing for bus error exceptions follows the standard exception processing sequence. Refer to **4.13 Exceptions** for more information. However, two special cases of bus error, called double bus faults, can abort exception processing.

$\overline{\text{BERR}}$ assertion is not detected until an instruction is complete. The $\overline{\text{BERR}}$ latch is cleared by the first instruction of the $\overline{\text{BERR}}$ exception handler. A double bus fault can occur in two ways:

1. When bus error exception processing begins, and a second $\overline{\text{BERR}}$ is detected before the first instruction of the exception handler is executed.
2. When one or more bus errors occur before the first instruction after a reset exception is executed.

Multiple bus errors within a single instruction that can generate multiple bus cycles cause a single bus error exception after the instruction has been executed.

Immediately after assertion of a second $\overline{\text{BERR}}$, the MCU halts and drives the $\overline{\text{HALT}}$ line low. Only a reset can restart a halted MCU; however, bus arbitration can still occur. Refer to **5.6.6 External Bus Arbitration** for more information. A bus error or address error that occurs after exception processing has been completed (during the execution of the exception handler routine, or later) does not cause a double bus fault.

5.6.5.3 Halt Operation

When $\overline{\text{HALT}}$ is asserted while $\overline{\text{BERR}}$ is not asserted, the MCU halts external bus activity after negation of $\overline{\text{DSACK}}$. The MCU may complete the current word transfer in progress. For a long-word to byte transfer, this could be after S2 or S4. For a word to byte transfer, activity ceases after S2.

Negating and reasserting $\overline{\text{HALT}}$ according to timing requirements provides single-step (bus cycle to bus cycle) operation. The $\overline{\text{HALT}}$ signal affects external bus cycles only, so that a program that does not use external bus can continue executing. During dynamically-sized 8-bit transfers, external bus activity may not stop at the next cycle boundary. Occurrence of a bus error while $\overline{\text{HALT}}$ is asserted causes the CPU16 to process a bus error exception.

When the MCU completes a bus cycle while the $\overline{\text{HALT}}$ signal is asserted, the data bus goes into a high-impedance state and the $\overline{\text{AS}}$ and $\overline{\text{DS}}$ signals are driven to their inactive states. The address, function code, size, and read/write lines remain in the same state.

The halt operation has no effect on bus arbitration. However, when external bus arbitration occurs while the MCU is halted, address and control signals go into a high-impedance state. If $\overline{\text{HALT}}$ is still asserted when the MCU regains control of the bus, the address, function code, size, and read/write lines revert to their previously driven states. The MCU cannot service interrupt requests while halted.

5.6.6 External Bus Arbitration

The MCU bus design provides for a single bus master at any one time. Either the MCU or an external device can be master. Bus arbitration protocols determine when an external device can become bus master. Bus arbitration requests are recognized during normal processing, $\overline{\text{HALT}}$ assertion, and when the CPU16 has halted due to a double bus fault.

The MCU bus controller manages bus arbitration signals so that the MCU has the lowest priority. External devices that need to obtain the bus must assert bus arbitration signals in the sequences described in the following paragraphs.

Systems that include several devices that can become bus master require external circuitry to assign priorities to the devices, so that when two or more external devices attempt to become bus master at the same time, the one having the highest priority becomes bus master first. The protocol sequence for assuming bus mastership from the MCU is:

1. An external device asserts the bus request signal ($\overline{\text{BR}}$).
2. The MCU asserts the bus grant signal ($\overline{\text{BG}}$) to indicate that the bus is available.
3. An external device asserts the bus grant acknowledge ($\overline{\text{BGACK}}$) signal to indicate that it has assumed bus mastership.

$\overline{\text{BR}}$ can be asserted during a bus cycle or between cycles. $\overline{\text{BG}}$ is asserted in response to $\overline{\text{BR}}$. To guarantee operand coherency, $\overline{\text{BG}}$ is only asserted at the end of operand transfer.

If more than one external device can be bus master, required external arbitration must begin when a requesting device receives $\overline{\text{BG}}$. An external device must assert $\overline{\text{BGACK}}$ when it assumes mastership, and must maintain $\overline{\text{BGACK}}$ assertion as long as it is bus master.

Two conditions must be met for an external device to assume bus mastership. The device must receive $\overline{\text{BG}}$ through the arbitration process, and $\overline{\text{BGACK}}$ must be inactive, indicating that no other bus master is active. This technique allows the processing of bus requests during data transfer cycles.

$\overline{\text{BG}}$ is negated a few clock cycles after $\overline{\text{BGACK}}$ transition. However, if bus requests are still pending after $\overline{\text{BG}}$ is negated, the MCU asserts $\overline{\text{BG}}$ again within a few clock cycles. This additional $\overline{\text{BG}}$ assertion allows external arbitration circuitry to select the next bus master before the current master has released the bus.

Refer to **Figure 5-19** which shows bus arbitration for a single device. The flow chart shows $\overline{\text{BR}}$ negated at the same time $\overline{\text{BGACK}}$ is asserted. Refer to the *SCIM Reference Manual* (SCIMRM/AD) for more information on bus arbitration.

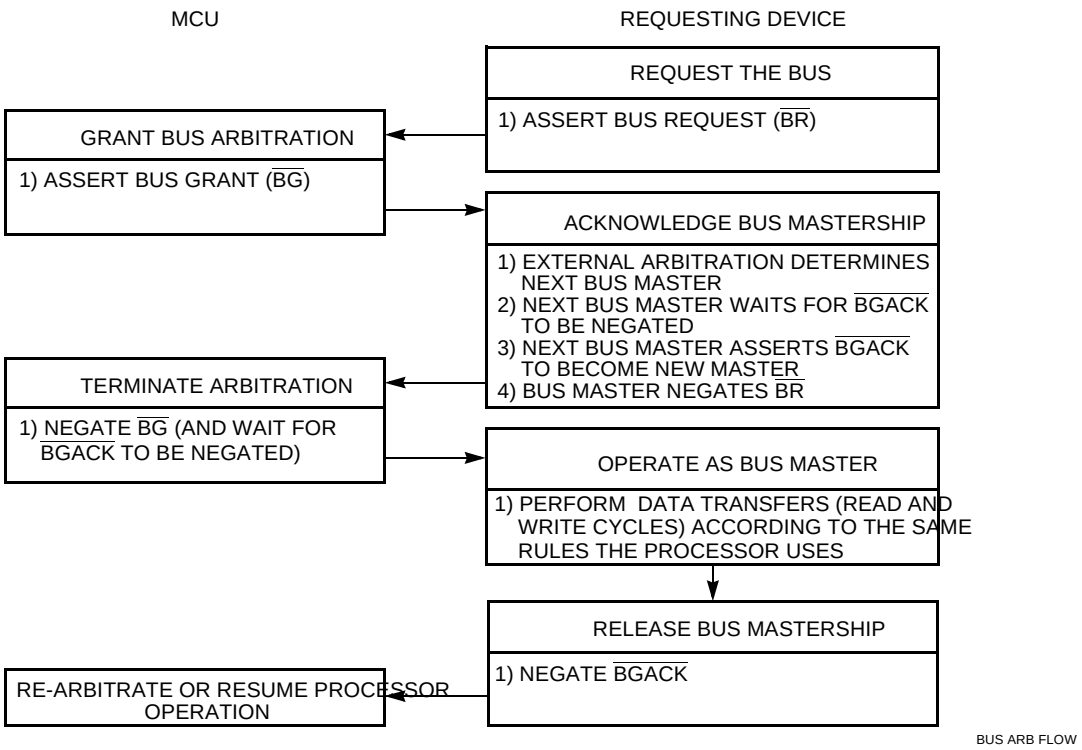

MCU
REQUESTING DEVICE
REQUEST THE BUS
1) ASSERT BUS REQUEST (BR)
GRANT BUS ARBITRATION
1) ASSERT BUS GRANT (BG)
ACKNOWLEDGE BUS MASTERSHIP
1) EXTERNAL ARBITRATION DETERMINES NEXT BUS MASTER
2) NEXT BUS MASTER WAITS FOR BGACK TO BE NEGATED
3) NEXT BUS MASTER ASSERTS BGACK TO BECOME NEW MASTER
4) BUS MASTER NEGATES BR
TERMINATE ARBITRATION
1) NEGATE BG (AND WAIT FOR BGACK TO BE NEGATED)
OPERATE AS BUS MASTER
1) PERFORM DATA TRANSFERS (READ AND WRITE CYCLES) ACCORDING TO THE SAME RULES THE PROCESSOR USES
RELEASE BUS MASTERSHIP
1) NEGATE BGACK
RE-ARBITRATE OR RESUME PROCESSOR OPERATION
BUS ARB FLOW

Figure 5-19 Bus Arbitration Flowchart for Single Request

5.6.6.1 Show Cycles

The MCU normally performs internal data transfers without affecting the external data bus, but it is possible to show these transfers during debugging. $\overline{AS}$ is not asserted externally during show cycles.

Show cycles are controlled by SHEN[1:0] in SCIMCR. This field set to %00 by reset. When show cycles are disabled, the address bus, function codes, size, and read/write signals reflect internal bus activity, but $\overline{AS}$ and $\overline{DS}$ are not asserted externally and external data bus pins are in a high-impedance state during internal accesses. Refer to **5.2.4 Show Internal Cycles** and the *SCIM Reference Manual* (SCIMRM/AD) for more information.

When show cycles are enabled, $\overline{DS}$ is asserted externally during internal cycles, and internal data is driven out on the external data bus. Because internal cycles normally continue to run when the external bus is granted, one SHEN[1:0] encoding halts internal bus activity while there is an external master.

The SIZ[1:0] signals reflect bus allocation during show cycles. Only the appropriate portion of the data bus is valid during the cycle. During a byte write to an internal address, the portion of the bus that represents the byte that is not written reflects internal bus conditions, and is indeterminate. During a byte write to an external address, the data multiplexer in the SCIM2 drives the value of the byte that is written to both halves of the data bus.

5.7 Reset

Reset occurs when an active low logic level on the $\overline{\text{RESET}}$ pin is clocked into the SCIM2. The $\overline{\text{RESET}}$ input is synchronized to the system clock. If there is no clock when $\overline{\text{RESET}}$ is asserted, reset does not occur until the clock starts. Resets are clocked to allow completion of write cycles in progress at the time $\overline{\text{RESET}}$ is asserted.

Reset procedures handle system initialization and recovery from catastrophic failure. The MCU performs resets with a combination of hardware and software. The SCIM2 determines whether a reset is valid, asserts control signals, performs basic system configuration and boot memory selection based on hardware mode-select inputs, then passes control to the CPU16.

5.7.1 Reset Exception Processing

The CPU16 processes resets as a type of asynchronous exception. An exception is an event that preempts normal processing, and can be caused by internal or external events. Exception processing makes the transition from normal instruction execution to execution of a routine that deals with an exception. Each exception has an assigned vector that points to an associated handler routine. These vectors are stored in the exception vector table. The exception vector table consists of 256 two-byte vectors and occupies 512 bytes of address space. The CPU16 uses vector numbers to calculate displacement into the table. Refer to **4.13 Exceptions** for more information.

Reset is the highest-priority CPU16 exception. Unlike all other exceptions, a reset occurs at the end of a bus cycle, and not at an instruction boundary. Handling resets in this way prevents write cycles in progress at the time the reset signal is asserted from being corrupted. However, any processing in progress is aborted by the reset exception, and cannot be restarted. Only essential reset tasks are performed during exception processing. Other initialization tasks must be accomplished by the exception handler routine. Refer to **5.7.3 Reset Processing Summary** for details on exception processing.

5.7.2 Reset Source Summary

SCIM2 reset control logic determines the cause of a reset, synchronizes request signals to CLKOUT, and asserts reset control logic. All resets are gated by CLKOUT. Asynchronous resets can occur on any clock edge and are assumed to be catastrophic. Synchronous resets are timed to occur at the end of bus cycles. When a synchronous reset is detected, the SCIM2 bus monitor is automatically enabled. If the bus cycle during which a synchronous reset is detected does not terminate normally, the bus monitor will terminate the cycle and allow the reset to proceed. **Table 5-19** is a summary of reset sources.

Table 5-19 Reset Source Summary

Type	Source	Timing
External	Assertion of $\overline{\text{RESET}}$ pin	Synchronous
Power on	Rising voltage on V_{DD}	Asynchronous
Software watchdog	Timeout of software watchdog	Asynchronous
Halt	Halt monitor (e.g. double bus fault)	Asynchronous
Loss of clock	Reference failure caught by loss of clock detector	Synchronous
Test	Test submodule	Synchronous

Internal byte and aligned word write cycles are guaranteed valid for synchronous resets. External writes will also complete uncorrupted, provided the data bus is conditioned with a circuit that incorporates $\overline{\text{RESET}}$, such as that shown in **Figure 5-21**.

5.7.3 Reset Processing Summary

Once SCIM2 reset control logic has synchronized an internal or external reset request, the internal $\overline{\text{RESET}}$ signal is asserted, and the following events take place:

1. Instruction execution is aborted.
2. The condition code register is initialized.
   a. The IP field is set to $7, disabling all interrupts below priority 7.
   b. The S bit is set, disabling LPSTOP mode.
   c. The SM bit is cleared, disabling MAC saturation mode.
3. The K register is cleared.

NOTE

All condition code register bits that are not initialized are not affected by reset; however, out of power-on reset, these bits are indeterminate.

The following events take place when the internal $\overline{\text{RESET}}$ signal is negated:

1. The CPU16 samples the $\overline{\text{BKPT}}$ input.
2. The CPU16 fetches reset vectors in the order that follows. Vectors can be fetched from on-chip memory or from an external memory device enabled by the $\overline{\text{CSBOOT}}$ signal.
   a. Initial ZK, SK, and PK extension field values
   b. Initial PC
   c. Initial SP
   d. Initial IZ value
3. The CPU16 begins fetching instructions pointed to by the initial PK:PC.

5.7.4 Reset Status Register

The reset status register (RSR) contains a bit for each reset source in the MCU. When a reset occurs, a bit corresponding to the reset type is set. When multiple causes of reset occur at the same time, more than one bit in RSR may be set. The reset status register is updated by the reset control logic when $\overline{\text{RESET}}$ is released. Refer to **APPENDIX D REGISTER SUMMARY**.

5.7.5 Reset Timing

When an external device asserts the $\overline{\text{RESET}}$ pin for at least four clock cycles, the signal will be latched and held internally until completion of the current bus cycle. Any further processing of the reset exception is then delayed until the SCIM2 reset control logic detects that the $\overline{\text{RESET}}$ pin is no longer being externally driven. Two clock cycles will elapse (during which time the pullup resistor on $\overline{\text{RESET}}$ will pull the pin high) while the reset control logic switches the $\overline{\text{RESET}}$ pin from an input to an output. $\overline{\text{RESET}}$ will then be driven low for 512 clock cycles.

If a synchronous internal reset is detected (e.g. from the loss of clock detector or the test submodule), the reset control logic will wait for bus cycle completion and then drive $\overline{\text{RESET}}$ low for 512 clock cycles. An asynchronous internal reset (e.g. from the halt monitor or the software watchdog) will immediately drive $\overline{\text{RESET}}$ low for 512 clock cycles without waiting for the current bus cycle to complete.

After the 512 clock cycle assertion of the $\overline{\text{RESET}}$ pin, the processing flow for both internal and external resets is the same. The SCIM2 reset control logic will release the $\overline{\text{RESET}}$ pin and read configuration information from $\overline{\text{BERR}}$, $\overline{\text{BKPT}}$, and DATA[15:0]. Refer to **5.7.8 Operating Configuration Out of Reset** for more information.

Ten clock cycles will elapse to allow the pullup resistor on $\overline{\text{RESET}}$ to pull the pin high. The reset control logic will then sample the $\overline{\text{RESET}}$ pin. If the pin is high, the reset control logic will release the external bus interface (EBI) and allow the reset vector to be fetched. If $\overline{\text{RESET}}$ is still low, 180 clock cycles will elapse, and the reset control logic will sample the pin again. As above, if $\overline{\text{RESET}}$ is high, processing will resume and the reset vector will be fetched.

If $\overline{\text{RESET}}$ still has not risen to logic one, the reset control logic will begin the external reset sequence as described at the beginning of this section. Further reset exception processing will not proceed until $\overline{\text{RESET}}$ is sampled at logic one after the 10 clock cycle or 180 clock cycle delays described above. **Figure 5-20** depicts the reset sequence for the SCIM2.

5.7.6 Power-On Reset

Power-on reset (POR) operation involves special circumstances related to the application of system power and, if the PLL is used, clock synthesizer power. V_{DD} ramp time affects pin state during reset. Slow V_{DD} ramp times can leave MCU pins in an indeterminate state longer than is desired or is tolerable in some applications.

When the PLL is used to generate the MCU system clock, oscillator start up time also determines how long MCU pins remain in an indeterminate state. Immediate application of V_{DDSYN}/MODCLK power and careful attention to crystal specifications and oscillator circuit design play an important role in minimizing start up time.

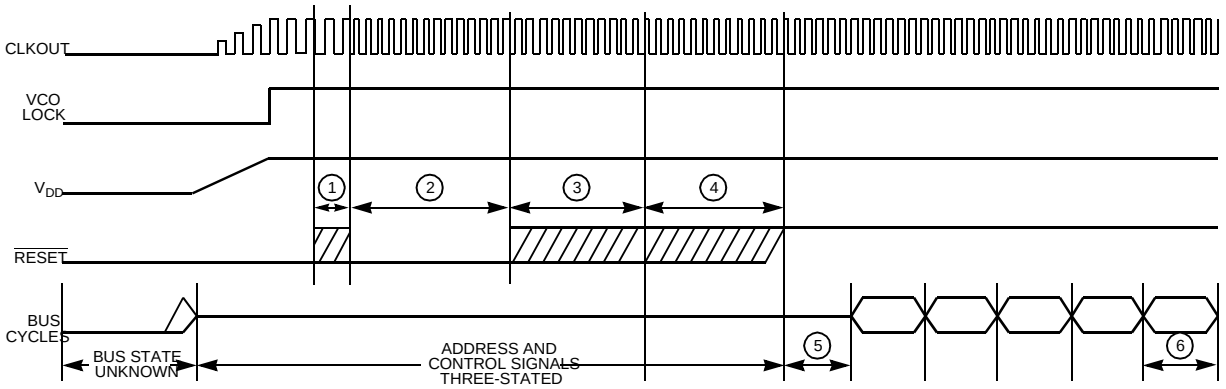
Grounding V_{DDSYN}/MODCLK places the MCU in external clock mode, initially operating at the frequency input on the EXTAL pin. In this case, any events requiring clock cycles during POR will occur as quickly as those clock cycles are input on the EXTAL pin.

Power-on reset activates a circuit in the SCIM2 that asserts the internal and external $\overline{\text{RESET}}$ lines. As V_{DD} ramps up to the minimum operating voltage, the PLL (if enabled) begins to generate the system clock and the internal $\overline{\text{RESET}}$ line is negated. This initializes SCIM2 pins the values shown in **Table 5-20**.

At this point, POR will proceed no further until the PLL locks at two or 256 times f_{ref} in fast or slow reference modes, respectively. Reset exception processing will then continue as outlined in **5.7.5 Reset Timing.** In external clock mode, the PLL is disabled which permits normal reset exception processing as soon as the internal $\overline{\text{RESET}}$ line is negated.

The SCIM2 propagates $\overline{\text{RESET}}$ and the system clock to all other MCU modules. Once the clock is running and internal $\overline{\text{RESET}}$ is asserted for at least four clock cycles, these modules reset. V_{DD} and PLL ramp up times determine how long these four clock cycles take. Worst case occurs in slow reference mode and is approximately 15 milliseconds. During this period, MCU pins may be in an indeterminate state. While pull-up resistors may be used on input only pins, active logic will be required to condition input/output or output only pins.

Figure 5-20 depicts the timing of the power-on reset sequence.



NOTES:
1. TWO CLOCK CYCLE DELAY REQUIRED BY RESET CONTROL LOGIC TO SWITCH RESET PIN FROM INPUT TO OUTPUT.
2. RESET CONTROL LOGIC DRIVES THE RESET PIN LOW FOR 512 CLOCK CYCLES TO GUARANTEE THIS LENGTH OF RESET TO THE ENTIRE SYSTEM.
3. TEN CLOCK CYCLE DELAY DURING WHICH THE RESET PIN IS NO LONGER DRIVEN AND IS ALLOWED TO RISE TO A LOGIC ONE. OPERATING CONFIGURAT LATCHED FROM DATA[15:0] AND BERR WHEN THIS DELAY BEGINS.
4. ADDITIONAL 180 CLOCK DELAY PROVIDED BY RESET CONTROL LOGIC TO ALLOW THE RESET PIN TO RISE TO A LOGIC ONE IF IT DID NOT DO SO DURING PRIOR 10 CLOCK CYCLES.
5. INTERNAL START-UP TIME.
6. FIRST INSTRUCTION FETCHED.

SCIM2 POR TIM

Figure 5-20 Power-On Reset

5.7.7 Pin State During Reset

It is important to keep the distinction between pin function and pin electrical state clear. Although control register values and mode select inputs determine pin function, a pin driver can be active, inactive, or in a high impedance state when reset occurs. During power-on reset, pin state is subject to the constraints discussed in **5.7.6 Power-On Reset**.

NOTE

Pins that are not used should either be configured as outputs (if possible) or be pulled to an appropriate inactive state. This decreases unnecessary current consumption caused by digital inputs floating near mid-supply level.

5.7.7.1 Reset States of SCIM2 Pins

Generally, while RESET is asserted, SCIM2 pins either go into a high-impedance state or are driven to their inactive states. Operating mode selection occurs when RESET is released, and mode select inputs must be driven to the appropriate states at this time. Use an active circuit, such as that shown in **Figure 5-21**, for this purpose. **Table 5-20** shows the state of SCIM2 pins during reset.

Table 5-20 SCIM2 Pin States During Reset

Pin(s)	Pin State During $\overline{\text{RESET}}$
ADDR[2:0]	High-Z
ADDR[10:3]/PB[7:0]	High-Z
ADDR[18:11]/PA]7:0]	High-Z
ADDR[22:19]/$\overline{\text{CS[9:6]}}$/PC[6:3]	V_{DD}
ADDR23/$\overline{\text{CS10}}$/ECLK	V_{DD}
$\overline{\text{AS}}$/PE5	High-Z
$\overline{\text{AVEC}}$/PE2	High-Z
$\overline{\text{BERR}}$	Mode Select Input
$\overline{\text{BG}}$/$\overline{\text{CSM}}$	V_{DD}
$\overline{\text{BGACK}}$/$\overline{\text{CSE}}$	V_{DD}
$\overline{\text{BR}}$/$\overline{\text{CS0}}$	V_{DD}
CLKOUT	Output
$\overline{\text{CSBOOT}}$	V_{DD}
DATA[7:0]/PH[7:0]	Mode Select Inputs
DATA[15:8]/PG[7:0]	Mode Select Inputs
$\overline{\text{DS}}$/PE4	High-Z
$\overline{\text{DSACK0}}$/PE0	High-Z
$\overline{\text{DSACK1}}$/PE1	High-Z
FASTREF/PF0	Mode Select Input
FC0/$\overline{\text{CS3}}$/PC0	V_{DD}
FC1/PC1	V_{DD}
FC2/$\overline{\text{CS3}}$/PC2	V_{DD}
$\overline{\text{HALT}}$	High-Z
$\overline{\text{IRQ[7:1]}}$/PF[7:1]	High-Z
PE3	High-Z
R/$\overline{\text{W}}$	High-Z
$\overline{\text{RESET}}$	Asserted
SIZ[1:0]/PE[7:6]	High-Z
TSC	Three State Enable Input

5.7.7.2 Reset States of Pins Assigned to Other MCU Modules

As a rule, module pins that can be configured for general purpose I/O go into a high-impedance state during reset. However, during power-on reset, module port pins may be in an indeterminate state for a short period of time. Refer to **5.7.6 Power-On Reset** for more information.

5.7.8 Operating Configuration Out of Reset

When $\overline{\text{RESET}}$ is released, the SCIM2 acquires setup information from several MCU pins. Individually or in groups, these pins control the four basic areas of MCU configuration outlined in **Table 5-21**.

Table 5-21 Pins Associated with Basic Configuration Options

Option	Controlling Pins
Background Debug Mode Disable/Enable	$\overline{\text{BKPT}}$
Flash/ROM Module Disable/Enable	DATA[15:12]
Operating Mode Selection	$\overline{\text{BERR}}$, DATA1
SCIM2 I/O Port Configuration	DATA[11:2], DATA0

Clock mode is not listed in **Table 5-21** because it is not selected when $\overline{\text{RESET}}$ is released. Instead, it is latched immediately from V_{DDSYN}/MODCLK and FASTREF/PF0 upon assertion of $\overline{\text{RESET}}$. Refer to **5.3.1 System Clock Sources** for more information.

5.7.8.1 Operating Mode Selection

The logic states of $\overline{\text{BERR}}$ and DATA1 determine MCU operating mode when $\overline{\text{RESET}}$ is released. Care should be taken to guarantee that $\overline{\text{BERR}}$ is driven to a known state during reset. Unlike DATA1 which has a weak pull-up resistor, no conditioning circuitry is present on the $\overline{\text{BERR}}$ pin. If $\overline{\text{BERR}}$ is allowed to float during reset, improper mode determination may occur. Operating mode selection is summarized in **Table 5-22**.

Table 5-22 Operating Mode Selection

$\overline{\text{BERR}}$	DATA1	Operating Mode
0	X[1]	Single Chip
1	0	16-Bit Expanded
1	1	8-Bit Expanded

NOTES:
1. When $\overline{\text{BERR}}$ is low, DATA1 has no effect on mode selection.

The configuration of the SCIM2 EBI is dependent on operating mode. ADDR[18:3] serve as general purpose I/O ports A and B when the MCU is running in single-chip mode. DATA[7:0] serve as general purpose I/O port H in the single-chip and 8-bit expanded modes, and DATA[15:8] serve as general purpose I/O port G in single-chip mode. **Table 5-23** summarizes I/O port availability of the address and data buses in each operating mode.

Table 5-23 Address and Data Bus I/O Port Availability

Operating Mode	Address Bus	Data Bus	I/O Ports
Single Chip	—	—	Port A = ADDR[18:11] Port B = ADDR[10:3] Port G = DATA[15:8] Port H = DATA[7:0]
8-Bit Expanded	ADDR[18:3]	DATA[15:8]	Port H = DATA[7:0]
16-Bit Expanded	ADDR[18:3]	DATA[15:0]	—

In single-chip mode, the default setting of the address bus disable (ABD) bit places ADDR[2:0] in a high-impedance state to reduce noise emissions. ADDR[2:0] function as normal address bus pins in expanded operating modes. Refer to **5.2.3 Noise Reduction in Single-Chip Mode** and **APPENDIX D REGISTER SUMMARY** for information on the address bus disable bit (ABD).

The ADDR[23:19] pins have multiple functions as high-order address lines, chip selects, or discrete outputs and are configured differently depending on operating mode selection and data bus conditioning when $\overline{\text{RESET}}$ is released. The following paragraphs contain a summary of pin configuration options for each external bus configuration.

5.7.8.2 Data Bus Mode Selection

DATA[15:0] have weak internal pull-up devices. When pins are held high by the internal pull-ups, the MCU uses a default operating configuration. Specific lines can be held low externally during reset to achieve alternate configurations.

NOTE

External bus loading can overcome the weak internal pull-up devices on data bus lines and hold pins low during reset.

Use an active device to hold data bus lines low. Data bus configuration logic must release the bus before the first bus cycle after reset to prevent conflict with external memory devices. The first bus cycle occurs ten CLKOUT cycles after $\overline{\text{RESET}}$ is released. If external mode selection logic causes a conflict of this type, an isolation resistor on the driven lines may be required. **Figure 5-21** shows a recommended method for conditioning the data bus mode select signals.

The mode configuration drivers are conditioned with R/$\overline{\text{W}}$ and $\overline{\text{DS}}$ to prevent conflicts between external devices and the MCU when $\overline{\text{RESET}}$ is asserted. If $\overline{\text{RESET}}$ is asserted during an external write cycle, R/$\overline{\text{W}}$ conditioning (as shown in **Figure 5-21**) prevents corruption of the data during the write. Similarly, $\overline{\text{DS}}$ conditions the mode configuration drivers so that external reads are not corrupted when $\overline{\text{RESET}}$ is asserted during an external read cycle.

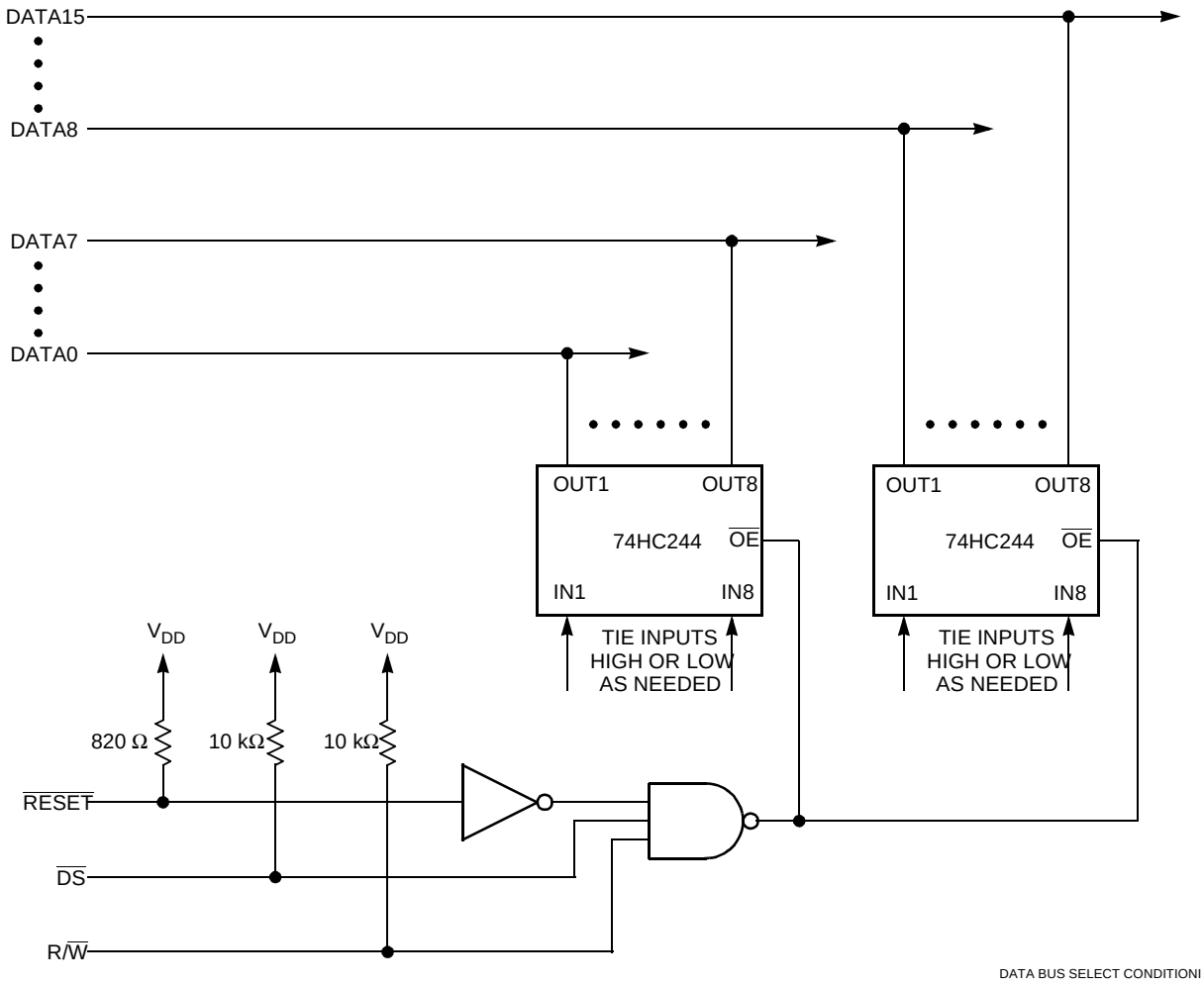


Figure 5-21 Preferred Circuit for Data Bus Mode Select Conditioning

Alternate methods can be used for driving data bus pins low during reset. **Figure 5-22** shows two of these options.

NOTE

These simpler circuits do not offer the protection from potential memory corruption during $\overline{\text{RESET}}$ assertion as does the circuit shown in **Figure 5-21**.

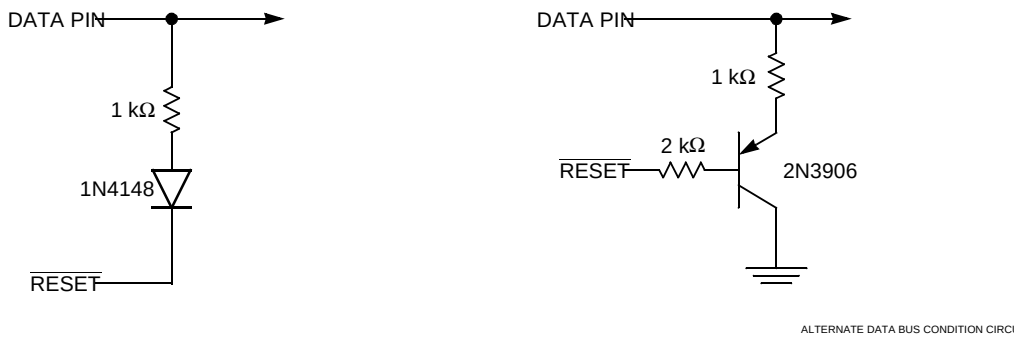


Figure 5-22 Alternate Circuit for Data Bus Mode Select Conditioning

In the simpler of these two circuits, a resistor is connected in series with a diode from the data bus pin to the RESET line. A bipolar transistor can be used for the same purpose, but an additional current limiting resistor must be connected between the base of the transistor and the RESET pin. If a MOSFET is substituted for the bipolar transistor, only the 1 kΩ isolation resistor is required.

5.7.8.3 Single-Chip Mode

When BERR = 0 at the release of RESET, single-chip operation is selected. BERR can be tied low to select this configuration at all times. The external bus interface is essentially disabled in single-chip mode, and SCIM2 pins generally serve as discrete inputs and outputs. The behavior of specific pin groups is discussed in the following paragraphs.

NOTE

The paragraphs that follow describe the behavior of SCIM2 pins in single-chip mode only. Sections that follow cover 16-bit and 8-bit expanded modes.

ADDR[2:0] have no discrete I/O function in single-chip mode. These pins are placed in a high-impedance state at power-on but can be enabled by clearing the ABD bit in SCIMCR.

ADDR[18:11] become port A input/output pins PA[7:0], and ADDR[10:3] become port B input/output pins PB[7:0]. Each port is configurable entirely as inputs or outputs on a per port basis by the DDA and DDB bits in the port A/B data direction register (DDRAB).

Special attention should be paid to chip-select pins in single-chip mode. While each chip-select base address register and option register is active and may be programmed as desired, a match condition will not assert the corresponding pin. For this reason, chip selects should be used expressly to provide autovector termination of interrupt acknowledge cycles generated in response to assertion of the IRQ[7:1] pins.

Because match conditions do not result in chip-select assertion, the %10 (8-bit port) and %11 (16-bit port) encodings of the pin assignment fields in CSPAR0 and CSPAR1 serve only to drive pins so configured high at all times. Consequently, any chip select may provide autovector termination, even if its pin assignment field in CSPAR0 or CSPAR1 is programmed with the %00 (discrete output) or %01 (alternate function) encoding.

The first chip select that should be used for autovector termination in single-chip mode is $\overline{\text{CSBOOT}}$; it has no discrete output or alternate function capability. Although typically not needed in single-chip mode, the SCIM2 bus arbitration feature may be used when the $\overline{\text{BR}}$/$\overline{\text{CS0}}$, $\overline{\text{BG}}$/$\overline{\text{CSM}}$, and $\overline{\text{BGACK}}$/$\overline{\text{CSE}}$ pins are configured for their alternate functions. Of these three pins, only $\overline{\text{BR}}$/$\overline{\text{CS0}}$ can provide autovector termination. The $\overline{\text{CSM}}$ and $\overline{\text{CSE}}$ chip selects function in emulation mode only and are not user programmable.

CSPAR0 initially configures the SCIM2 function code pins FC[2:0] as port C discrete outputs PC[2:0]. Each pin may, however, still operate as a function code output, and when configured as such, will be driven during appropriate bus cycles. The chip-select functions of FC0/$\overline{\text{CS3}}$/PC0 and FC2/$\overline{\text{CS5}}$/PC2 may also be used for autovector termination in the fashion described above.

ADDR[22:19]/$\overline{\text{CS[9:6]}}$/PC[6:3] are initially configured as port C discrete outputs PC[6:3] by chip-select pin assignment register 1 (CSPAR1). Each pin may, however, still operate as an address line, and when configured as such, will be driven during appropriate bus cycles.

CSPAR1 initially configures ADDR23/$\overline{\text{CS10}}$/ECLK as a 16-bit chip select (%11 pin assignment field encoding) to drive the pin to its inactive state. ADDR23/$\overline{\text{CS10}}$/ECLK has no discrete output function. When %00 is programmed into its pin assignment field in CSPAR1, ADDR23/$\overline{\text{CS10}}$/ECLK will drive the M6800 bus E clock signal.

Just as the chip-select, function code, and bus arbitration signals associated with port C can be made active in single-chip mode, so too can the bus control signals associated with port E. While initially configured as discrete I/O by the port E pin assignment register (PEPAR), any port E bus control signal can be made active and will be driven or accept input during appropriate bus cycles.

Port F pins will initially be configured for discrete I/O in single-chip mode but can otherwise serve as interrupt request lines or edge-detect I/O pins with optional interrupt capability. Because no external bus is available in single-chip mode, interrupt requests from port F pins configured as interrupts (as opposed to interrupt requests from the port F edge-detect logic) must have their interrupt acknowledge cycles terminated by autovector.

In single-chip mode, the data bus is disabled at all times. DATA[15:8] become port G input/output pins PG[7:0], and DATA[7:0] become port H input/output pins PH[7:0]. Port G and H pins are configurable as inputs or outputs on a per pin basis.

Like ADDR[2:0] (which can be disabled by setting the ABD bit in SCIMCR), the R/$\overline{\text{W}}$ line and instruction tracking pins (IPIPE0/DSO and IPIPE1/DSI) can be disabled by setting RWD and CPUD bits in SCIMCR, respectively.

5.7.8.4 Fully (16-bit) Expanded Mode

Operation in 16-bit expanded mode is selected when $\overline{BERR}$ = 1 and DATA1 = 0 at the release of $\overline{RESET}$. In this configuration, ADDR[18:11]/PA[7:0] and ADDR[10:3]/PB[7:0] become part of the address bus. Likewise, DATA[15:8]/PG[7:0] and DATA[7:0]/PH[7:0] become the data bus. The ABD, RWD, and CPUD bits in SCIMCR are clear, enabling ADDR[2:0], R/$\overline{W}$, and the instruction tracking pins (IPIPE0/DSO and IPIPE1/DSI), respectively. Ports A, B, G, and H are unavailable in 16-bit expanded mode. The initial configuration of all other SCIM2 pins is controlled by DATA[11:2] and DATA0 and is outlined in **Table 5-24** below.

Table 5-24 Fully (16-bit) Expanded Mode Reset Configuration

Select Pin	Affected Pin(s)	Default Function (Pin Held High)	Alternate Function (Pin Held Low)
DATA0	$\overline{CSBOOT}$	16-bit $\overline{CSBOOT}$	8-bit $\overline{CSBOOT}$
DATA2	$\overline{BR}$/$\overline{CS0}$ FC0/$\overline{CS3}$/PC0 FC1/PC1 FC2/$\overline{CS5}$/PC2	$\overline{CS0}$ $\overline{CS3}$ FC1 $\overline{CS5}$	$\overline{BR}$ FC0 FC1 FC2
DATA[7:3]	ADDR23/$\overline{CS10}$/ECLK ADDR[22:19]/$\overline{CS}$[9:6]/PC[6:3]	See **Table 5-25**	
DATA8	$\overline{DSACK0}$/PE0 $\overline{DSACK1}$/PE1 $\overline{AVEC}$/PE2 $\overline{RMC}$/PE3[1] $\overline{DS}$/PE4 $\overline{AS}$/PE5 SIZ0/PE6 SIZ1/PE7	$\overline{DSACK0}$ $\overline{DSACK1}$ $\overline{AVEC}$ $\overline{RMC}$ $\overline{DS}$ $\overline{AS}$ SIZ0 SIZ1	PE0 PE1 PE2 PE3 PE4 PE5 PE6 PE7
DATA9	FASTREF/PF0 $\overline{IRQ}$[7:1]/PF[7:1]	FASTREF[2] $\overline{IRQ}$[7:1]	PF0 PF[7:1]
DATA10[3]	$\overline{BGACK}$/$\overline{CSE}$ $\overline{BG}$/$\overline{CSM}$	$\overline{BGACK}$ $\overline{BG}$	$\overline{CSE}$ $\overline{CSM}$[4]
DATA11[5]	External Bus Interface	Normal Operation	Factory Test

NOTES:

1. Like all other port E pins, PE3 has an associated bus control signal, $\overline{RMC}$. On CPU32-based MCUs only, assertion of $\overline{RMC}$ indicates the execution of an indivisible read-modify-write bus cycle. Because $\overline{RMC}$ will never be asserted on CPU16-based MCUs, the pin can be configured for PE3 discrete I/O use at all times without any loss of external bus interface functionality.
2. The FASTREF function is used only at reset and serves no purpose during normal operation.
3. If DATA1 and DATA10 are low at the rising edge of $\overline{RESET}$, the SCIM2 will operate in emulation mode, a special variation of 16-bit expanded mode.
4. For $\overline{CSM}$ to be active, the SCIM2 must be configured for emulation mode, as described above, and any on-chip masked ROM modules must be disabled by driving their associated data bus pins low at the rising edge of $\overline{RESET}$. At present, only masked ROM modules support memory emulation by means of the $\overline{CSM}$ chip select. $\overline{CSM}$ will not assert on MCUs with flash EEPROM modules.
5. DATA11 must be high at the rising edge of $\overline{RESET}$ for normal MCU operation.

DATA[7:3] select in a contiguous fashion whether ADDR[23:19]/$\overline{CS}$[10:6] serve as high-order address lines or chip selects. **Table 5-25** shows the reset correspondence between these pins.

Table 5-25 Reset Pin Function of $\overline{CS[10:6]}$

Data Bus Pins at Reset					Chip-Select/Address Bus Pin Function				
DATA7	DATA6	DATA5	DATA4	DATA3	$\overline{CS10}$/ ADDR23	$\overline{CS9}$/ ADDR22	$\overline{CS8}$/ ADDR21	$\overline{CS7}$/ ADDR20	$\overline{CS8}$/ ADDR19
1	1	1	1	1	$\overline{CS10}$	$\overline{CS9}$	$\overline{CS8}$	$\overline{CS7}$	$\overline{CS6}$
1	1	1	1	0	$\overline{CS10}$	$\overline{CS9}$	$\overline{CS8}$	$\overline{CS7}$	ADDR19
1	1	1	0	X	$\overline{CS10}$	$\overline{CS9}$	$\overline{CS8}$	ADDR20	ADDR19
1	1	0	X	X	$\overline{CS10}$	$\overline{CS9}$	ADDR21	ADDR20	ADDR19
1	0	X	X	X	$\overline{CS10}$	ADDR22	ADDR21	ADDR20	ADDR19
0	X	X	X	X	ADDR23	ADDR22	ADDR21	ADDR20	ADDR19

DATA[15:12] allow implementation dependent disabling of on-chip ROM and/or flash EEPROM modules. **Table 5-26** shows which modules on M68HC16 R-series MCUs are affected by these pins.

Table 5-26 Reset Configuration for M68HC16 R-Series Memory Modules

Select Pin	State of Select Pin at Rising Edge of $\overline{RESET}$	MC68HC16R1	MC68HC916R1	MC68HC16R3	MC68HC916R3
DATA12[1]	0	NA[2]	2K BEFLASH array disabled	NA	2K BEFLASH array disabled
	1		2K BEFLASH array enabled		2K BEFLASH array enabled
DATA14[3]	0	48K Masked ROM array disabled	16K and 32K FLASH arrays disabled	32K and 64K Masked ROM arrays disabled	All Three 32K FLASH arrays disabled
	1	48K Masked ROM array enabled	16K and 32K FLASH arrays enabled	32K and 64K Masked ROM arrays enabled	All three 32K FLASH arrays enabled

NOTES:

1. The BEFLASH array is disabled if its STOP shadow bit is programmed to one or if DATA12 is low at the rising edge of $\overline{RESET}$. The BEFLASH module is not present on the MC68HC16R1/R3.
2. NA = no affect on reset configuration.
3. The 16K and 32K flash arrays operate in a fashion similar to the BEFLASH array and can be disabled if their STOP shadow bits are programmed to one or if DATA14 is low at the rising edge of $\overline{RESET}$. The masked ROM arrays are enabled or disabled out of reset solely based upon the state of DATA14; they do not have STOP shadow bits.

5.7.8.5 Partially (8-bit) Expanded Mode

Operation in 8-bit expanded mode is selected when $\overline{BERR}$ = 1 and DATA1 = 1 at the release of $\overline{RESET}$. In this configuration, ADDR[18:11]/PA[7:0] and ADDR[10:3]/PB[7:0] become part of the address bus, and only DATA[15:8]/PG[7:0] are used for the data bus. The ABD, RWD, and CPUD bits in SCIMCR are clear, enabling ADDR[2:0], R/$\overline{W}$, and the instruction tracking pins (IPIPE0/DSO and IPIPE1/DSI), respectively. Ports A, B, and G are unavailable in 8-bit expanded mode, and DATA[7:0]/PH[7:0] serve as port H discrete I/O pins only. All remaining SCIM2 pins are configured as shown in **Table 5-27.**

Table 5-27 Partially (8-bit) Expanded Mode Reset Configuration

Select Pin	Affected Pin(s) or Module(s)	Default Function (Pin Held High)	Alternate Function (Pin Held Low)
NA[1]	$\overline{\text{CSBOOT}}$	8-bit $\overline{\text{CSBOOT}}$	
NA[1]	$\overline{\text{BR}}$/$\overline{\text{CS0}}$ FC0/$\overline{\text{CS3}}$/PC0 FC1/PC1 FC2/$\overline{\text{CS5}}$/PC2	$\overline{\text{CS0}}$ $\overline{\text{CS3}}$ FC1 $\overline{\text{CS5}}$	
NA[1]	ADDR23/$\overline{\text{CS10}}$/ECLK ADDR[22:19]/$\overline{\text{CS}}$[9:6]/PC[6:3]	$\overline{\text{CS}}$[10:6]	
DATA8	$\overline{\text{DSACK0}}$/PE0 $\overline{\text{DSACK1}}$/PE1 $\overline{\text{AVEC}}$/PE2 $\overline{\text{RMC}}$/PE3[1] $\overline{\text{DS}}$/PE4 $\overline{\text{AS}}$/PE5 SIZ0/PE6 SIZ1/PE7	$\overline{\text{DSACK0}}$ $\overline{\text{DSACK1}}$ $\overline{\text{AVEC}}$ $\overline{\text{RMC}}$ $\overline{\text{DS}}$ $\overline{\text{AS}}$ SIZ0 SIZ1	PE0 PE1 PE2 PE3 PE4 PE5 PE6 PE7
DATA9	FASTREF/PF0 $\overline{\text{IRQ}}$[7:1]/PF[7:1]	FASTREF[2] $\overline{\text{IRQ}}$[7:1]	PF0 PF[7:1]
DATA10	$\overline{\text{BGACK}}$/$\overline{\text{CSE}}$[3] $\overline{\text{BG}}$/$\overline{\text{CSM}}$[3]	$\overline{\text{BGACK}}$ $\overline{\text{BG}}$	

NOTES:
1. Because DATA[7:0] are unavailable in 8-bit expanded mode, these pins default to the reset configurations noted.
2. The FASTREF function is used only at reset and serves no purpose during normal operation.
3. The $\overline{\text{CSE}}$ and $\overline{\text{CSM}}$ emulation chip selects do not function in 8-bit expanded mode.

Just as in 16-bit expanded mode, DATA[15:12] allow implementation dependent disabling of on-chip ROM and/or flash EEPROM modules. Refer to **Table 5-26** above for which modules on M68HC16 R-series MCUs are affected by these pins.

5.7.8.6 Breakpoint Mode Selection

Background debug mode (BDM) is enabled when the breakpoint ($\overline{\text{BKPT}}$) pin is sampled at logic zero at the release of $\overline{\text{RESET}}$. Subsequent assertion of the $\overline{\text{BKPT}}$ pin or the internal breakpoint signal (for instance, execution of the CPU16 BGND instruction) will place the CPU16 in BDM.

If $\overline{\text{BKPT}}$ is sampled at logic one at the rising edge of $\overline{\text{RESET}}$, BDM is disabled. Assertion of the $\overline{\text{BKPT}}$ pin will result in normal breakpoint exception processing. Execution of the BGND instruction will cause an illegal instruction exception to be taken.

BDM remains enabled until the next system reset. $\overline{\text{BKPT}}$ is relatched and synchronized on each rising transition of $\overline{\text{RESET}}$ and must be held low for at least two clock cycles prior to $\overline{\text{RESET}}$ negation for BDM to be enabled. $\overline{\text{BKPT}}$ assertion logic must be designed with special care. If $\overline{\text{BKPT}}$ assertion extends into the first bus cycle following the release of $\overline{\text{RESET}}$, the bus cycle could inadvertently be tagged with a breakpoint.

Refer to **4.14.4 Background Debug Mode** and the *CPU16 Reference Manual* (CPU16RM/AD) for more information on background debug mode. Refer to the *SCIM Reference Manual* (SCIMRM/AD) and APPENDIX A ELECTRICAL CHARACTERISTICS for more information concerning BKPT signal timing.

5.7.8.7 Emulation Mode Selection

The SCIM2 contains logic that can be used to replace on-chip ports externally. The SCIM2 also contains special support logic that allows external emulation of internal ROM. These emulation support features enable the development of a single-chip application in expanded mode.

Emulation mode is a special type of 16-bit expanded operation. It is entered by holding DATA10 low, BERR high, and DATA1 low during reset. In emulation mode, all port A, B, E, G, and H data and data direction registers and the port E pin assignment register are mapped externally. The port C data, port F data and data direction registers, and port F pin assignment register are accessible normally in emulation mode.

The port emulation chip select (CSE) is asserted whenever any of the externally mapped registers are addressed. The signal is asserted on the falling edge of AS. The SCIM2 does not respond to these accesses, allowing external logic, such as a port replacement unit (PRU) to respond. Accesses to externally mapped registers require three clock cycles.

ROM emulation is enabled by holding BERR high and by holding low DATA1, DATA10, and any of DATA[15:12] required to disable on-chip masked ROM modules when RESET is released. While ROM emulation mode is enabled, the emulation memory chip-select signal (CSM) is asserted whenever an access to an address assigned to a masked ROM module is made.

ROM modules do not acknowledge IMB accesses while in emulation mode. This causes the SCIM2 to run an external bus cycle for each access.

NOTE

Flash EEPROM modules do not support emulation mode. If ROM emulation is enabled, the CSM chip-select will remain driven high when accesses are made to flash EEPROM modules.

5.7.9 Use of the Three-State Control Pin

Asserting the three-state control (TSC) input causes the MCU to place all output drivers in a disabled, high-impedance state. TSC must remain asserted for approximately ten clock cycles in order for drivers to change state.

When the SCIM2 clock synthesizer is used, PLL ramp-up time affects how long the ten cycles take. Worst case is approximately 20 milliseconds from TSC assertion.

When an external clock signal is applied, pins go high-impedance as soon after TSC assertion as approximately ten clock pulses have been applied to the EXTAL pin.

NOTE

When TSC assertion takes effect, internal signals are forced to values that can cause inadvertent mode selection. Once the output drivers change state, the MCU must be powered down and restarted before normal operation can resume.

5.8 Interrupts

Interrupt recognition and servicing involve complex interaction between the SCIM2, the CPU16, and a device or module requesting interrupt service. This discussion provides an overview of the entire interrupt process. Chip-select logic can also be used to respond to interrupt requests. Refer to **5.9 Chip-Selects** for more information.

5.8.1 Interrupt Exception Processing

The CPU16 handles interrupts as a type of asynchronous exception. An exception is an event that preempts normal processing. Exception processing makes the transition from normal instruction execution to execution of a routine that deals with an exception. Each exception has an assigned vector that points to an associated handler routine. These vectors are stored in a vector table located from $00000 to $001FF. The CPU16 uses vector numbers to calculate displacement into the table. Refer to **4.13 Exceptions** for more information.

5.8.2 Interrupt Priority and Recognition

The CPU16 provides seven levels of interrupt priority (1 - 7), seven automatic interrupt vectors, and 200 user-assignable interrupt vectors. All interrupts with priorities less than seven can be masked by the interrupt priority (IP) field in the condition code register (CCR).

The IP field consists of CCR bits [7:5]. Binary values %000 to %111 provide eight priority masks. Each mask prevents an interrupt request of a priority less than or equal to the mask value (except for $\overline{\text{IRQ7}}$) from being recognized. When the IP field contains %000, no interrupt is masked. During exception processing, the IP field is set to the priority of the interrupt being serviced.

There are seven interrupt request signals ($\overline{\text{IRQ[7:1]}}$) with corresponding external pins that can be asserted by microcontroller modules or external devices. Simultaneous requests of different priorities can be made. Internal assertion of an interrupt request line does not affect the state of the corresponding MCU pin.

External interrupt requests are routed to the CPU16 via the EBI and SCIM2 interrupt control logic. All requests for interrupt service are treated as if they come from internal modules. The CPU16 treats external interrupt requests as if they come from the SCIM2.

The $\overline{\text{IRQ[6:1]}}$ pins are active-low level-sensitive inputs. The $\overline{\text{IRQ7}}$ pin is an active-low transition-sensitive input; it requires both an edge and a voltage level to be valid.

$\overline{IRQ[6:1]}$ are maskable. $\overline{IRQ7}$ is non-maskable. The $\overline{IRQ7}$ input is transition sensitive to prevent redundant servicing and stack overflow. A non-maskable interrupt is generated each time $\overline{IRQ7}$ is asserted and each time the CCR is written while $\overline{IRQ7}$ is asserted. A write to the CCR re-arms the $\overline{IRQ7}$ detection circuitry; consequently, any write to the CCR while $\overline{IRQ7}$ is asserted, even one that sets the IP field to %111, will generate a new $\overline{IRQ7}$ interrupt.

Interrupt requests are sampled on consecutive falling edges of the system clock. Interrupt request input circuitry has hysteresis. To be valid, a request signal must be asserted for at least two consecutive clock cycles. Valid requests do not cause immediate exception processing, but are left pending. Pending requests are processed at instruction boundaries or when processing of higher priority exceptions is complete.

The CPU16 does not latch the priority of pending interrupt requests. If an interrupt source of higher priority makes a request while a lower priority request is pending, the higher priority request will be serviced. If an interrupt request with a priority less than or equal to the current IP mask value is made, the CPU16 will not recognize the request. If simultaneous interrupt requests of different priorities are made, and both have a priority greater than the mask value, the CPU16 will recognize the higher priority request.

5.8.3 Interrupt Acknowledge and Arbitration

When the CPU16 detects one or more interrupt requests of a priority higher than the IP mask value, a read cycle from address %11111111111111111111: [IP]:1 in CPU space is executed. Refer to **5.5.1.7 Function Codes** and **5.6.4 CPU Space Cycles** for more information.

The CPU space read cycle performs to functions. It places a mask value corresponding to the highest priority interrupt request on the address bus, and it acquires an exception vector number from the interrupt source. The mask value is decoded by modules or external devices that have requested interrupt service to determine whether the current interrupt acknowledge (IACK) cycle pertains to them. It is also latched into the IP field to mask lower priority interrupts during exception processing.

Modules that have requested interrupt service decode the IP value placed on the address bus at the beginning of the IACK cycle, and if their requests are at the specified IP level, respond to the cycle. Arbitration between simultaneous requests of the same priority is performed by serial contention between module interrupt arbitration (IARB) field bit values.

Each module that can request interrupt service, including the SCIM2, has an IARB field in its module configuration register. To implement an arbitration scheme, each module that can request interrupt service must be assigned a unique, non-zero IARB field value during system initialization. Arbitration priorities can range from %0001 (lowest) to %1111 (highest). If the CPU16 recognizes an interrupt request from a module that has an IARB field value of %0000, a spurious interrupt exception will be processed.

Because the EBI manages external interrupt requests, the SCIM2 IARB field value is used for arbitration between internal and external interrupt requests of the same priority. The reset value of IARB for the SCIM2 is %1111, and the reset value of IARB for all other modules is %0000. As noted above, initialization software must assign different values to each IARB field to implement an arbitration scheme.

NOTE

Do not assign the same arbitration priority to more than one module. When two or more IARB fields have the same nonzero value, the CPU16 will interpret multiple vector numbers simultaneously with unpredictable consequences.

Although arbitration is intended to deal with simultaneous interrupt requests of the same priority level, it always take place, even when a single source is requesting service. This is important for two reasons: the EBI does not transfer the IACK cycle to the external bus unless the SCIM2 wins contention, and failure to contend causes the IACK cycle to be terminated early by bus error.

When arbitration is complete, the winning module must place a vector number on the data bus and terminate the IACK cycle with $\overline{\text{DSACK}}$. In the case of external interrupt requests, the IACK cycle is transferred to the external bus. The device requesting interrupt service must decode the mask value then respond with a vector number and generate data and size acknowledge ($\overline{\text{DSACK}}$) termination signals, or it must assert $\overline{\text{AVEC}}$ to request an autovector. If the device does not respond in time, the SCIM2 bus monitor, if enabled, will assert the internal $\overline{\text{BERR}}$ signal, and a spurious interrupt exception will be taken.

Chip-select logic can also be used to generate internal $\overline{\text{AVEC}}$ or $\overline{\text{DSACK}}$ signals in response to external interrupt requests. Chip-select address match logic functions only after the SCIM2 has won arbitration, and the resulting IACK cycle is transferred to the external bus. For this reason, interrupt requests from modules other than the SCIM2 will never have their IACK cycles terminated by chip-select generated $\overline{\text{AVEC}}$ or $\overline{\text{DSACK}}$. Refer to **5.9.3 Using Chip-Select Signals for Interrupt Acknowledge Cycle Termination** for more information.

As stated above, all interrupt requests from internal modules have their associated IACK cycles terminated by $\overline{\text{DSACK}}$. For this reason, user vectors (instead of autovectors) must always be used for interrupts generated by internal modules.

For periodic timer interrupts, the PIRQL[2:0] field in the periodic interrupt control register (PICR) determines PIT priority level. A PIRQL[2:0] value of %000 disables PIT interrupts. By hardware convention, PIT interrupts are serviced before external interrupt service requests or port F edge-detect interrupts requests of the same priority. External interrupt requests are serviced before requests from the port F edge-detect logic, as well. Refer to **5.4.6 Periodic Interrupt Timer** and **5.10.3 Port F** for more information.

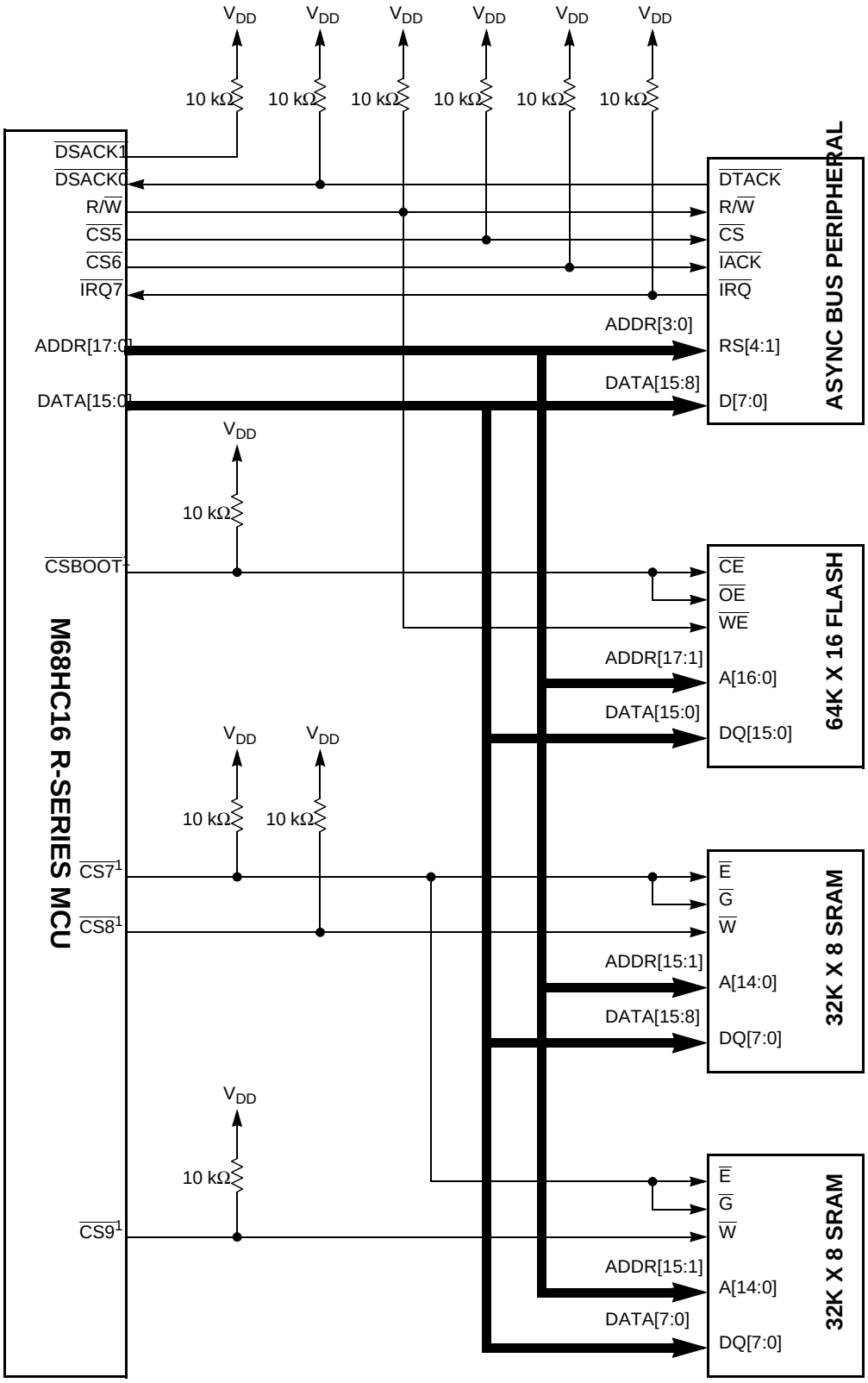
5.8.4 Interrupt Processing Summary

A summary of the entire interrupt processing sequence follows. When the sequence begins, a valid interrupt service request has been detected and is pending.

1. The CPU16 finishes higher priority exception processing or reaches an instruction boundary.
2. Processor state is stacked, then the CCR PK extension field is cleared.
3. The interrupt acknowledge cycle begins:
   a. FC[2:0] are driven to %111 (CPU space) encoding.
   b. The address bus is driven as follows. ADDR[23:20] = %1111 ADDR[19:16] = %1111, which indicates that the cycle is an interrupt acknowledge CPU space cycle; ADDR[15:4] = %111111111111; ADDR[3:1] = the priority of the interrupt request being acknowledged; and ADDR0 = %1.
   c. Request priority is latched into the CCR IP field from the address bus.
4. Modules or external peripherals that have requested interrupt service decode the priority value in ADDR[3:1]. Each module or device with a request level equal to the value in ADDR[3:1] enters interrupt arbitration.
5. After arbitration, the interrupt acknowledge cycle is completed in one of the following ways:
   a. When there is no contention (responding modules have IARB = %0000), the spurious interrupt monitor asserts $\overline{\text{BERR}}$, and the CPU16 generates the spurious interrupt vector number.
   b. The interrupt source that wins arbitration supplies a vector number and $\overline{\text{DSACK}}$ signals appropriate to the access. The CPU16 acquires the vector number.
   c. The $\overline{\text{AVEC}}$ signal is asserted either by the external device requesting interrupt service ($\overline{\text{AVEC}}$ can be tied low if all external interrupts are to use autovectors) or by an appropriately programmed SCIM2 chip select, and the CPU16 generates an autovector number corresponding to the interrupt priority.
   d. The bus monitor or external device asserts $\overline{\text{BERR}}$ and the CPU16 generates the spurious interrupt vector number.
6. The vector number is converted to a vector address.
7. The content of the vector address is loaded into the PC and the processor transfers control to the exception handler routine.

5.9 Chip-Selects

Typical microcontrollers require additional hardware to provide chip-select signals for external devices. The SCIM2 includes 9 programmable chip-select circuits that can provide from 2 to 16 clock cycle access to external memory and peripherals. Address block sizes of 2 Kbytes to 1 Mbyte can be selected. Because ADDR[23:20] follow the state of ADDR19 on the CPU16, 512 Kbytes is the largest usable block size on CPU16-based MCUs. **Figure 5-23** is a diagram of a basic system that uses chip-selects.



NOTES:
1. THESE CHIP-SELECT LINES ARE CONFIGURED FOR 16-BIT PORT OPERATION IN THIS EXAI

Figure 5-23 Basic MCU System

Chip-select assertion can be synchronized with bus control signals to provide output enable, read/write strobe, or interrupt acknowledge signals. Chip-select logic can also generate DSACK and AVEC signals internally. Each signal can also be synchronized with the ECLK signal available on ADDR23.

When a memory access occurs, chip-select logic compares address space type, address, type of access, transfer size, and interrupt priority (in the case of interrupt acknowledge) to parameters stored in chip-select registers. If all parameters match, the appropriate chip-select signal is asserted. Select signals are active low. If a chip-select function is given the same address as a microcontroller module or an internal memory array, an access to that address goes to the module or array, and the chip-select signal is not asserted. The external address and data buses do not reflect the internal access.

All chip-select circuits are configured for operation out of reset. However, all chip-select signals except CSBOOT are disabled, and cannot be asserted until the R/W[1:0] and BYTE[1:0] fields in the corresponding option register are programmed to non-zero values. CSBOOT is automatically asserted out of reset in 8-bit and 16-bit expanded modes. Alternate functions for chip-select pins are enabled if appropriate data bus pins are held low at the release of RESET. Refer to **5.7.8.2 Data Bus Mode Selection** for more information. **Figure 5-24** is a functional diagram of a single chip-select circuit.

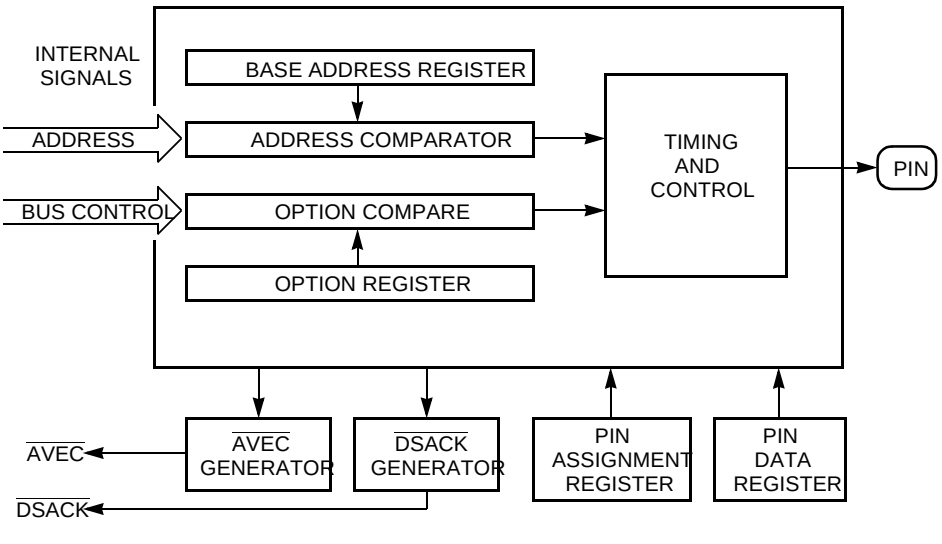


Figure 5-24 Chip-Select Circuit Block Diagram

5.9.1 Chip-Select Registers

Each chip-select pin can have one or more functions. Chip-select pin assignment registers CSPAR[1:0] determine the functions of chip-select pins. The pin assignment registers also determine port size (8- or 16-bit) for dynamic bus allocation. A pin data register (PORTC) latches data for chip-select pins that are used as discrete outputs.

Blocks of address space are assigned to each chip select. Block sizes of 2 Kbytes to 1 Mbyte can be selected by writing to the BLKSZ[2:0] field in each chip-select base address register (CSBAR[10:5], CSBAR3, CSBAR0, CSBARBT). Because ADDR[23:20] always follow the the state of ADDR19 on the CPU16, 512 Kbytes is the largest usable block size on CPU16-based MCUs. Multiple chip-selects assigned to the same block of addresses must have the same number of wait states.

Chip-select option registers (CSOR[10:5], CSOR3, CSOR0, CSORBT) determine timing of and conditions for assertion of chip-select signals. Eight parameters, including operating mode, access size, synchronization, and wait state insertion can be specified.

Initialization software usually resides in a peripheral memory device controlled by the chip-select circuits. A a dedicated boot chip-select ($\overline{\text{CSBOOT}}$) and associated registers (CSORBT and CSBARBT) are provided to support bootstrap operation.

Comprehensive address maps and register diagrams are provided in **APPENDIX D REGISTER SUMMARY**.

5.9.1.1 Chip-Select Pin Assignment Registers

Each pin assignment registers contain twelve 2-bit fields that determine the functions of the chip-select pins. Each pin has two or three possible functions, as shown in **Table 5-28**.

Table 5-28 Chip-Select Pin Functions

Chip-Select	Alternate Function	Discrete Output
$\overline{\text{CSBOOT}}$	$\overline{\text{CSBOOT}}$	—
$\overline{\text{CS0}}$	$\overline{\text{BR}}$	—
$\overline{\text{CSM}}$	$\overline{\text{BG}}$	—
$\overline{\text{CSE}}$	$\overline{\text{BGACK}}$	—
$\overline{\text{CS3}}$	FC0	PC0
—	FC1	PC1
$\overline{\text{CS5}}$	FC2	PC2
$\overline{\text{CS6}}$	ADDR19	PC3
$\overline{\text{CS7}}$	ADDR20	PC4
$\overline{\text{CS8}}$	ADDR21	PC5
$\overline{\text{CS9}}$	ADDR22	PC6
$\overline{\text{CS10}}$	ADDR23	ECLK

Table 5-29 shows pin assignment field encoding. Pins that have no discrete output function must not be programmed with the %00 encoding as this will configure the pin for its alternate function. For instance, programming CS0PA[1:0] to %00 will configure $\overline{\text{CS0}}$/$\overline{\text{BR}}$ for the bus request ($\overline{\text{BR}}$) function.

Table 5-29 Pin Assignment Field Encoding

CSxPA[1:0]	Description
00	Discrete output
01	Alternate function
10	Chip-select (8-bit port)
11	Chip-select (16-bit port)

Port size determines the way in which bus transfers to an external address are allocated. A port size of eight bits or sixteen bits can be selected when a pin is assigned as a chip select. Port size and transfer size affect how the chip-select signal is asserted. Refer to **5.9.1.3 Chip-Select Option Registers** for more information.

Out of reset, chip-select pin functions are determined by logic levels on certain data bus pins. The data bus pins have weak internal pull-up devices but can be held low by external logic. This allows a pin's 16-bit chip-select function (data bus pin(s) held high) or its alternate function (data bus pin(s) held low) to be selected at the release of RESET. Refer to **5.7.8.2 Data Bus Mode Selection** for more information.

The CSBOOT signal is enabled out of reset. The state of DATA0 during reset determines what port width CSBOOT uses. If DATA0 is held high, 16-bit port size is selected. If DATA0 is held low, 8-bit port size is selected. In 8-bit expanded mode, the state of DATA0 is ignored, and CSBOOT is configured for 8-bit operation.

A pin programmed as a discrete output will drive the value specified in the port C data register. No discrete output function is available for the CSBOOT, CS0/BR, CSM/BG,and CSE/BGACK pins. ADDR23 provides the ECLK output rather than a discrete output signal.

When a pin is programmed for discrete output or alternate function, internal chip-select logic still functions and can be used to generate DSACK or AVEC (to terminate IACK cycles generated in response to external interrupt requests) internally on an address and control signal match.

5.9.1.2 Chip-Select Base Address Registers

Each chip-select has an associated base address register. A base address is the lowest address in the block of addresses enabled by a chip-select. Block size is the extent of the address block above the base address. Block size is determined by the value contained in BLKSZ[2:0]. Multiple chip-selects assigned to the same block of addresses must have the same number of wait states.

BLKSZ[2:0] determines which bits in the base address field are compared to corresponding bits on the address bus during an access. Provided other constraints determined by option register fields are also satisfied, when a match occurs, the associated chip-select signal is asserted. **Table 5-30** shows BLKSZ[2:0] encoding.

Table 5-30 Block Size Encoding

BLKSZ[2:0]	Block Size	Address Lines Compared
000	2 Kbytes	ADDR[23:11]
001	8 Kbytes	ADDR[23:13]
010	16 Kbytes	ADDR[23:14]
011	64 Kbytes	ADDR[23:16]
100	128 Kbytes	ADDR[23:17]
101	256 Kbytes	ADDR[23:18]
110	512 Kbytes	ADDR[23:19]
111	512 Kbytes[1]	ADDR[23:20]

NOTES:
1. On CPU16-based MCUs, ADDR[23:10] always follow the state of ADDR19, thus limiting the maximum usable block size to 512 Kbytes.

The chip-select address compare logic uses only the most significant bits to match an address within a block. For this reason, the value of the base address must be an integer multiple of the block size.

Because ADDR[23:20] always follow the state of ADDR19 on CPU16-based MCUs, the maximum block size if 512 Kbytes. This also requires that the ADDR[23:20] bits in each chip select base address be programmed to the same state as the ADDR19 bit in order for match conditions to occur.

IMB addresses $080000 to $F7FFFF are inaccessible and will never be generated on CPU16-based MCUs. The CPU16's contiguous 1 Mbyte memory map from $00000 to $FFFFF appears on the IMB (and consequently the external bus) from $000000 to $07FFFF and from $F80000 to $FFFFFF. Refer to **3.5 CPU16 Memory Mapping** for more information.

After reset, the CPU16 fetches initialization values from addresses $00000 to $00007 in program space. To support bootstrap operation from reset, the chip-select boot base address register (CSBARBT) defaults to $0007 which specifies a base address of $000000 and a block size of 512 Kbytes. This allows a memory device containing the reset vector and startup routine to automatically be selected by $\overline{\text{CSBOOT}}$. Refer to **5.9.4 Chip-Select Reset Operation** for more information.

5.9.1.3 Chip-Select Option Registers

Fields in the chip-select option registers determine the timing of and conditions for assertion of chip-select signals. For a chip select to assert and to provide $\overline{\text{DSACK}}$ or autovector termination, other constraints set by fields in its option register and base address register must also be satisfied. The following paragraphs summarize option register functions. Refer to **APPENDIX D REGISTER SUMMARY** for register and bit field information.

The MODE bit determines whether chip-select assertion is asynchronous or is synchronized to the M6800-type bus clock signal ECLK available on ADDR23. Refer to **5.3 System Clock** for more information on ECLK.

The BYTE field controls the data placement conditions under which a particular chip select asserts. This is a different function from that of the chip-select pin assignment registers which determine if transfers controlled by a particular chip select are fundamentally eight or sixteen bits in length. Instead, BYTE[1:0] specifies whether the chip select will assert for data placed on the lower half, upper half, or both halves of the data bus.

When a chip select is configured for 8-bit port operation, only DATA[15:8] are used. Consequently, any BYTE field value other than %00 will permit signal assertion when all other match conditions are met.

When a chip select is configured for 16-bit port operation, BYTE[1:0] determines which combinations of ADDR0 and SIZ0 will result in chip-select assertion. A chip select configured for both bytes (%11) will assert (assuming all other conditions are met) regardless of the states of ADDR0 and SIZ0. A chip select configured for upper byte (%10) will assert only when ADDR0 = 0 (even addresses). A chip select configured for lower byte (%01) must assert on all accesses to odd addresses (ADDR0 = 1) and on word accesses to even addresses (ADDR0 = 0 and SIZ0 = 1). When the boolean expression $\overline{ADDR0} \cdot SIZ0$ is false, lower byte chip-select assertion will occur. In all cases, the routing of information onto the data bus by the EBI data multiplexer is controlled by ADDR0 and SIZ0.

R/$\overline{W}$[1:0] causes a chip-select signal to be asserted only for reads, only for writes, or for both reads and writes. Use this field in conjunction with the STRB bit to generate asynchronous control signals for external devices.

The STRB bit controls the timing of chip-select assertion in asynchronous mode. Selecting address strobe causes a chip-select signal to be asserted synchronized to the address strobe ($\overline{AS}$) signal. Selecting data strobe causes a chip-select signal to be asserted synchronized to the data strobe ($\overline{DS}$) signal. This bit has no effect in synchronous mode.

$\overline{DSACK[3:0]}$ specifies the source of $\overline{DSACK}$ in asynchronous mode. It also allows the user to optimize bus speed in a particular application by controlling the number of wait states that are inserted.

NOTE

The $\overline{DSACK[1:0]}$ pins are always active.

SPACE[1:0] determines the address space in which a chip-select is asserted. An access must have the space type specified by the SPACE[1:0] encoding in order for a chip-select signal to be asserted.

When SPACE[1:0] = %00 (CPU space), IPL[2:0] specifies the interrupt priority that must be matched when chip-select logic is used to terminate IACK cycles generated

in response to external requests for interrupt service. When SPACE[1:0] is set to %00 (CPU space), ADDR[3:1] is compared to the IPL field at the beginning of an IACK cycle. If these values are the same (and other option register constraints are satisfied), the specified chip select will be asserted. This field only affects the response of chip-select logic to IACK cycles and does not affect interrupt recognition by the CPU16. Setting IPL[2:0] to %000 when SPACE[1:0] = %00 will cause chip-select assertion regardless of the IACK cycle priority, provided other option register conditions are met. When SPACE[1:0] = %01, %10, or %11, the IPL field specifies whether chip-select assertion should occur during accesses to data space, program space, or both.

The $\overline{\text{AVEC}}$ bit is used to make a chip select respond to an interrupt acknowledge cycle. If the $\overline{\text{AVEC}}$ bit is set, an autovector will be selected for the particular external interrupt being serviced. If $\overline{\text{AVEC}}$ is zero, the interrupt acknowledge cycle will be terminated with $\overline{\text{DSACK}}$, and a vector number must be supplied by the external device requesting interrupt service.

5.9.1.4 Port C Data Register

The port C data register (PORTC) latches data for port C pins programmed as discrete outputs. When a pin is assigned as a discrete output, the value in this register appears at the output. Port C bit 7 is not used. Writing to this bit has no effect, and it always reads zero.

5.9.2 Chip-Select Operation

When the MCU makes an access, each enabled chip-select circuit compares the following items:

- Function code signals FC[2:0] to the SPACE field, and to the IP field if the SPACE field is not programmed for CPU space.
- Appropriate address bus bits to base address field.
- The R/$\overline{\text{W}}$ signal to the R/$\overline{\text{W}}$ field.
- ADDR0 and/or SIZ to the BYTE field (only chip selects configured for 16-bit operation).
- Priority of the interrupt being acknowledged (ADDR[3:1]) to the IPL field when the access is an interrupt acknowledge cycle and the SPACE field is programmed for CPU space.

When a match occurs, the chip-select signal is asserted. Assertion occurs at the same time as $\overline{\text{AS}}$ or $\overline{\text{DS}}$ assertion in asynchronous mode. Assertion is synchronized with ECLK in synchronous mode. In asynchronous mode, the $\overline{\text{DSACK}}$ field specifies internal or external $\overline{\text{DSACK}}$ assertion and the number of wait states inserted if internal $\overline{\text{DSACK}}$ assertion is selected.

The number of wait states needed by an external device is determined by its access time. Normally, wait states are inserted into the bus cycle during state S3 until a peripheral asserts $\overline{\text{DSACK}}$. If a peripheral does not generate $\overline{\text{DSACK}}$, internal $\overline{\text{DSACK}}$ generation must be selected and a predetermined number of wait states can be programmed into the chip-select option register. Refer to the *SCIM Reference Manual* (SCIMRM/AD) for further information.

5.9.3 Using Chip-Select Signals for Interrupt Acknowledge Cycle Termination

Ordinary bus cycles are issued in supervisor or user space, but interrupt acknowledge bus cycles are issued in CPU space. Refer to **5.6.4 CPU Space Cycles** and **5.8 Interrupts** for more information. The SCIM2 chip selects operate identically in each type of space, but base address and option registers must be properly programmed for each type of external bus cycle.

During a CPU space cycle, bits [15:3] of the appropriate base register must be configured to match ADDR[23:11], as the address is compared to an address generated by the CPU. Address lines [23:20] follow the state of ADDR19 on the CPU16, consequently, chip-select base address register bits [15:12] must be programmed to the same state as bit 11.

Figure 5-25 shows CPU space encoding for an interrupt acknowledge cycle. FC[2:0] drive %111, designating a CPU space access. ADDR[3:1] denote interrupt priority, and the space type field (ADDR[19:16]) is set to %1111, the interrupt acknowledge code. The rest of the address lines are set to one.

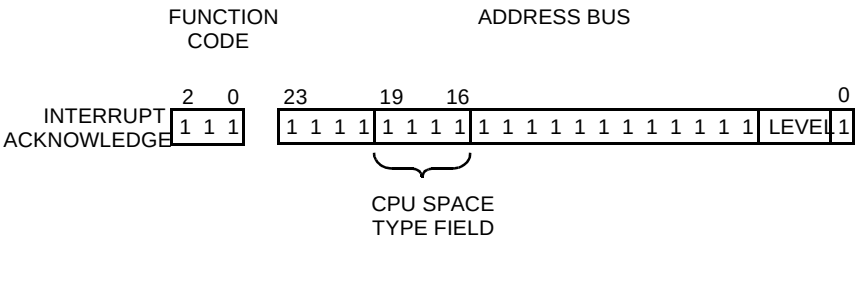

FUNCTION CODE
ADDRESS BUS
2 0
23 19 16 0
INTERRUPT ACKNOWLEDGE
1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 LEVEL 1
CPU SPACE TYPE FIELD
CPU SPACE IACK TII

Figure 5-25 CPU Space Encoding for Interrupt Acknowledge

Chip-select address match logic functions only after the SCIM2 has won arbitration, and the resulting IACK cycle is transferred to the external bus. For this reason, interrupt requests from modules other than the SCIM2 will never have their IACK cycles terminated by chip-select generated $\overline{\text{AVEC}}$ or $\overline{\text{DSACK}}$.

Use the procedure that follows to configure a chip select to provide IACK cycle termination.

1. Program the base address field to all ones.
2. Program block size to no more than 64 Kbytes, so that the address comparator checks ADDR[19:16] against the corresponding bits in the base address register. (The CPU space bus cycle type is placed on ADDR[19:16]).
3. Set the R/$\overline{\text{W}}$ field to read only. An interrupt acknowledge cycle is performed as a read in CPU space.
4. Set the BYTE field to lower byte when using a 16-bit port, as the external vector for a 16-bit port is fetched from the lower byte. Set the BYTE field to upper byte when using an 8-bit port.

If an interrupting device does not provide a vector number, an autovector must be generated, either by asserting the AVEC pin or by having the chip select assert AVEC internally. The latter is accomplished by setting the chip-select option register AVEC bit. This terminates the bus cycle.

5.9.4 Chip-Select Reset Operation

The LSB of each of the 2-bit pin assignment fields in CSPAR0 and CSPAR1 has a reset value of one. The reset values of the MSBs of each field are determined by the states of DATA[7:1] during reset. Weak internal pull-up devices condition each of the data lines so that chip-select operation is selected by default out of reset. Excessive bus loading can overcome the internal pull-up devices, resulting in inadvertent configuration out of reset. Use external pull-up resistors or active devices to avoid this.

The base address fields in chip-select base address registers CSBAR[10:5], CSBAR3, and CSBAR0 and chip-select option registers CSOR[10:5], CSOR3, and CSOR0 have the reset values shown in **Table 5-31**. The BYTE and R/W fields of each option register have a reset value of “disable”, so that a chip-select signal cannot be asserted until the base and option registers are initialized.

Table 5-31 Chip-Select Base and Option Register Reset Values

Fields	Reset Values
Base address	$000000
Block size	2 Kbytes
Async/sync Mode	Asynchronous mode
Upper/lower byte	Disabled
Read/write	Disabled
AS/DS	AS
DSACK	No wait states
Address space	CPU space
IPL	Any level
Autovector	External interrupt vector

Following reset, the MCU fetches initialization values from the reset vector, beginning at $000000 in supervisor program space. The CSBOOT chip-select signal is enabled and can select an external boot device mapped to a base address of $000000.

The MSB of the CSBTPA field in CSPAR0 has a reset value of one, so that chip-select function is selected by default out of reset. The BYTE field in chip-select option register CSORBT has a reset value of “both bytes” so that the select signal is enabled out of reset. The LSB of the CSBOOT field, determined by the logic level of DATA0 during reset, selects the boot ROM port size. When DATA0 is held low during reset, a port size of eight bits is selected. When DATA0 is held high during reset, a port size of 16 bits is selected. DATA0 has a weak internal pull-up device, so that a 16-bit port is selected by default out of reset. As mentioned above, the internal pull-up device can be overcome by bus loading effects. To ensure a particular configuration out of reset, use a pull-up resistor or an active device to place DATA0 in a known state during reset.

The base address field in the boot chip-select base address register CSBARBT has a reset value of all zeros, so that when the initial access to address $000000 is made, an address match occurs, and the $\overline{\text{CSBOOT}}$ signal is asserted. The block size field in CSBARBT has a reset value of %111 (1 Mbyte on CPU32-based MCUs and 512 Kbytes on CPU16-based MCUs). **Table 5-32** shows $\overline{\text{CSBOOT}}$ reset values.

Table 5-32 $\overline{\text{CSBOOT}}$ Base and Option Register Reset Values

Fields	Reset Values
Base address	$000000
Block size	1 Mbyte[1]
Async/sync mode	Asynchronous mode
Upper/lower byte	Both bytes
Read/write	Read/write
$\overline{\text{AS}}$/$\overline{\text{DS}}$	$\overline{\text{AS}}$
$\overline{\text{DSACK}}$	13 wait states
Address space	Supervisor space
IPL	Any level
Autovector	External vector externally

NOTES:
1. On CPU16-based MCUs, ADDR[23:20] always follow the state of ADDR19, thus limiting the maximum block size to 512 Kbytes.

5.10 General-Purpose Input/Output

The SCIM2 has six general-purpose input/output ports: A, B, E, F, G, and H. (Port C, an output-only port, is included under the discussion of chip-selects). Ports A, B, and G are available in single-chip mode only and port H is available in single-chip and 8-bit expanded modes only. Ports E, F, G, and H have associated data direction registers to configure each port pin as an input or output. Ports A and B share a data direction register that configures each port entirely as inputs or outputs. Ports E and F have associated pin assignment registers that allow the digital I/O or alternate function of each port pin to be selected. Port F has an edge-detect flag register that indicates whether a transition has occurred on any of its pins.

Table 5-33 shows the shared functions of the general-purpose I/O ports and the modes in which they are available.

Table 5-33 General-Purpose I/O Ports

Port	Shared Function	Modes
A	ADDR[18:11]	Single-chip
B	ADDR[10:3]	Single-chip
E	Bus control signals	All
F	$\overline{\text{IRQ}}$[7:1]/FASTREF	All
G	DATA[15:8]	Single-chip
H	DATA[7:0]	Single-chip, 8-bit expanded

Access to the port A, B, E, G, and H data and data direction registers, and the port E pin assignment register require three clock cycles to ensure timing compatibility with external port replacement logic. Accesses to the port F registers require two clock cycles. Port registers are byte-addressable and are grouped to allow coherent word access to port data register pairs A-B and G-H, as well as word-aligned long word coherency of the port A-B-G-H data registers.

If emulation mode is enabled, accesses to the port A, B, F, G, and H data and data direction registers and the port E pin assignment register are mapped externally, and cause the $\overline{\text{CSE}}$ port emulation chip select to be asserted. The SCIM2 does not respond to these accesses, but allows external logic, such as the Motorola MC68HC33 port replacement unit (PRU), to respond. Accesses to the port F registers will still be handled by the SCIM2.

A write to the port A, B, E, F, G, or H data register is stored in the port's internal data latch. If any port pin is configured as an output, the value stored for that bit is driven on the pin. A read of the port data register returns the value at the pin only if the pin is configured as a discrete input. Otherwise, the value read is the value stored in the data latch.

5.10.1 Ports A and B

Ports A and B are available in single-chip mode only. One data direction register controls the data direction for both ports. The SCIM2 will respond to port A and B registers at any time the MCU is not in emulation mode.

The port A/B data direction bits (DDA and DDB) control the direction of the pin drivers for ports A and B, respectively. Setting DDA or DDB to one configures all corresponding port pins as outputs. Clearing DDA or DDB to zero configures all corresponding port pins as inputs.

5.10.2 Port E

Port E can be made available in all operating modes. The state of $\overline{\text{BERR}}$ and DATA8 at the release of $\overline{\text{RESET}}$ controls whether the port E pins are initially configured as bus control signals or discrete I/O lines.

If the MCU is in emulation mode, accesses to the port E data, data direction, and pin assignment registers (PORTE, DDRE, and PEPAR) are mapped externally. This allows port replacement logic to be supplied externally, giving an emulator access to the bus control signals.

The port E data register (PORTE) is a single register that can be accessed in two locations ($YFFA11 and $YFFA13). The SCIM2 will respond to PORTE accesses at any time the MCU is not in emulation mode.

The port E data direction register (DDRE) controls the direction of the pin drivers when port E pins are configured for I/O. Any bit in this register set to one configures the corresponding pin as an output. Any bit in this register cleared to zero configures the corresponding pin as an input. The SCIM2 will respond to DDRE accesses at any time the MCU is not in emulation mode.

Port E pin assignment register (PEPAR) bits control the function of each port E pin. Any bit set to one configures the corresponding pin as a bus control signal with the function shown in **Table 5-34**. Any bit cleared to zero configures the corresponding pin as an I/O pin controlled by PORTE and DDRE.

Table 5-34 Port E Pin Assignments

PEPAR Bit	Port E Signal	Bus Control Signal
PEPA7	PE7	SIZ1
PEPA6	PE6	SIZ0
PEPA5	PE5	$\overline{AS}$
PEPA4	PE4	$\overline{DS}$
PEPA3	PE3	$\overline{RMC}$[1]
PEPA2	PE2	$\overline{AVEC}$
PEPA1	PE1	$\overline{DSACK1}$
PEPA0	PE0	$\overline{DSACK0}$

NOTES:
1. $\overline{RMC}$ is a CPU32 control signal. The CPU16 does not use $\overline{RMC}$, thus $\overline{RMC}$/PE3 can be used for digital I/O at all times.

$\overline{BERR}$ and DATA8 control the state of PEPAR following reset. If $\overline{BERR}$ and/or DATA8 are low during reset, PEPAR defaults to $00, configuring all port E pins as I/O pins. If $\overline{BERR}$ and DATA8 are both high during reset, PEPAR defaults to $FF, which configures all port E pins as bus control signals.

5.10.3 Port F

Port F consists of eight I/O pins, a data register, a data direction register, a pin assignment register, an edge-detect flag register, an edge-detect interrupt vector register, an edge-detect interrupt level register, and associated control logic. **Figure 5-26** is a block diagram of port F pins, registers, and control logic.

Port F pins can be configured as interrupt request inputs, edge-detect input/outputs, or discrete input/outputs. When port F pins are configured for edge detection, and a priority level is specified in the port F edge-detect interrupt level register (PFLVR), the port F control logic will generate an interrupt request when the specified edge is detected. Interrupt vector assignment is made by writing a value to the port F edge-detect interrupt vector register (PFIVR). Edge-detect interrupts have the lowest arbitration priority in the SCIM2.

A write to the port F data register (PORTF) is stored in the internal data latch, and if any port F pin is configured as an output, the value stored for that bit is driven on the pin. A read of PORTF returns the value on a pin only if the pin is configured as a discrete input. Otherwise, the value read is the value stored in the data latch. PORTF is a single register that can be accessed in two locations ($YFFA19 and $YFFA1B). It can be read or written at any time, and is not affected when the MCU is in emulation mode.

The port F data direction register (DDRF) controls the direction of the pin drivers when port F pins are configured for I/O. Any bit in this register set to one configures the corresponding pin as an output. Any bit in this register cleared to zero configures the corresponding pin as an input.

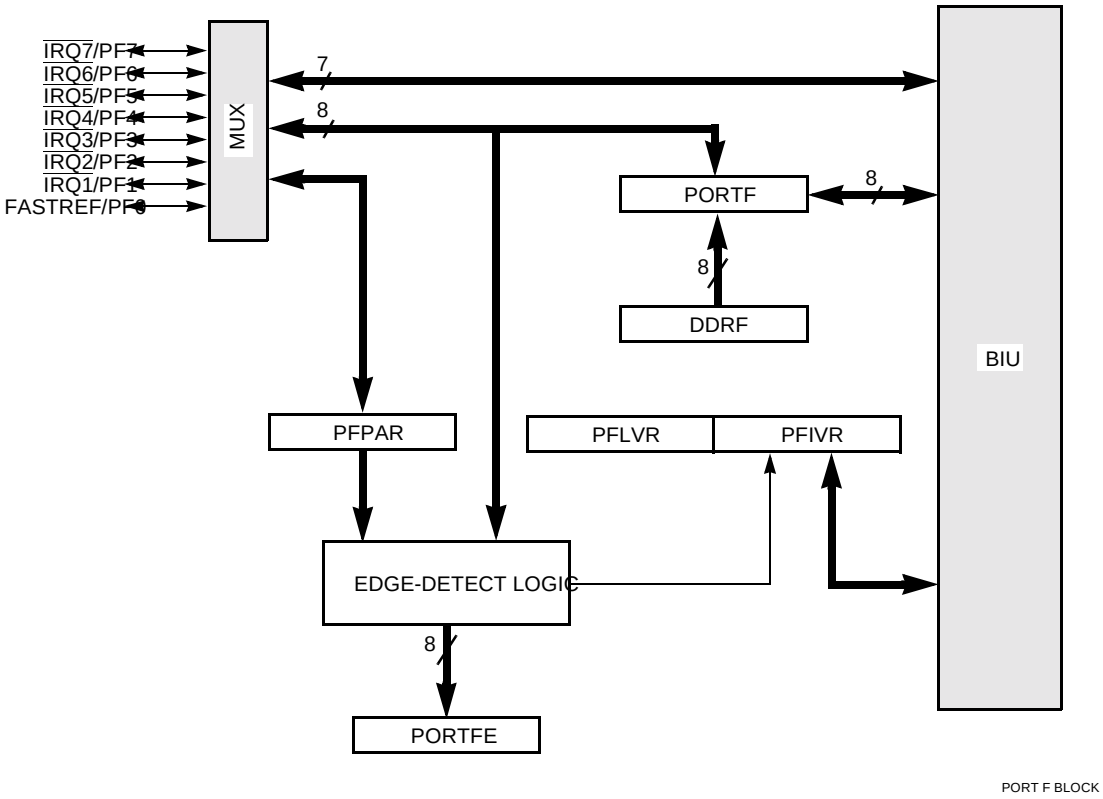


Figure 5-26 Port F Block Diagram

Port F pin assignment register (PFPAR) fields determine the functions of pairs of port F pins. **Table 5-35** shows port F pin assignments. **Table 5-36** shows PFPAR pin functions.

$\overline{\text{BERR}}$ and DATA9 control the state of PFPAR following reset. If $\overline{\text{BERR}}$ and/or DATA9 are low during reset, PFPAR defaults to $00, configuring all port F pins as I/O pins. If $\overline{\text{BERR}}$ and DATA9 are both high during reset, PFPAR defaults to $FF, which configures all port F pins as interrupt request inputs (except for FASTREF/PF0 which will be configured as FASTREF).

Table 5-35 Port F Pin Assignments

PFPAR Field	Port F Signal	Alternate Signal
PFPA3	PF[7:6]	$\overline{\text{IRQ}}$[7:1]
PFPA2	PF[5:4]	$\overline{\text{IRQ}}$[5:4]
PFPA1	PF[3:2]	$\overline{\text{IRQ}}$[3:2]
PFPA0	PF[1:0]	$\overline{\text{IRQ1}}$, FASTREF

Table 5-36 PFPAR Pin Functions

PFPAx[1:0]	Port F Signal
00	I/O pin without edge detect
01	Rising edge detect
10	Falling edge detect
11	Interrupt request

When a port F pin is configured for edge detection, a corresponding bit is set in the port F edge-detect flag register (PORTFE) when the specified edge is detected. PORTFE bits remain set, regardless of the subsequent state of the corresponding pin, until cleared. To clear a bit, first read PORTFE, then write the bit to zero. When a port F pin is configured for general-purpose I/O or for use as an interrupt request input, PORTFE bits do not change state.

The port F edge-detect interrupt vector register (PFIVR) determines which vector in the exception vector table is used for interrupt requests generated by the port F edge-detect logic. The port F edge-detect interrupt level register (PFLVR) determines the priority level of port F edge-detect interrupts. The reset value is $00, indicating that port F edge-detect interrupts are disabled.

5.10.4 Port G

Port G is available in single-chip mode only. These pins are always configured for use as general-purpose I/O in single-chip mode.

The SCIM2 will respond to port G data register (PORTG) accesses at any time the MCU is not in emulation mode. Reset has no effect on this register.

The port G data direction register (DDRG) controls the direction of the pin drivers when port G pins are configured for I/O. Setting a bit configures the corresponding pin as an output. Clearing a bit configures the corresponding pin as an input.

5.10.5 Port H

Port H is available in single-chip and 8-bit expanded modes only. The function of these pins is determined by the operating mode. There is no pin assignment register associated with this port.

The SCIM2 will respond to port H data register (PORTH) accesses at any time the MCU is not in emulation mode. Reset has no effect on this register.

The port H data direction register (DDRH) controls the direction of the pin drivers when port H pins are configured for I/O. Setting a bit configures the corresponding pin as an output. Clearing a bit configures the corresponding pin as an input.

5.11 Factory Test

The test submodule supports scan-based testing of the various MCU modules. It is integrated into the SCIM2 to support production test. Test submodule registers are intended for Motorola use only. Register names and addresses are provided in **APPENDIX D REGISTER SUMMARY** to show the user that these addresses are occupied. The QUOT pin is also used for factory test.

SECTION 6
STANDBY RAM MODULE

The standby RAM (SRAM) module consists of a fixed-location control register block and an array of fast (two clock) static RAM that may be mapped to a user specified location in the system memory map. The MC68HC16R1 and the MC68HC916R1 have a 2-Kbyte array. The MC68HC16R3 and the MC68HC916R3 have a 4-Kbyte array. The SRAM can be mapped to any address that is an integer multiple of the array size so long as the SRAM array does not overlap other memory arrays or module control registers. Overlap will result in array and/or register inaccessibility and must be avoided. Data can be read/written in bytes, words or long words. SRAM is powered by V_{DD} in normal operation. During power-down, SRAM contents can be maintained by power from the V_{STBY} input. Power switching between sources is automatic. A power down status bit denotes the loss of V_{STBY} during normal operation and when V_{DD} is powered down.

6.1 SRAM Register Block

There are four SRAM control registers: the RAM module configuration register (RAMMCR), the RAM test register (RAMTST), and the RAM array base address registers (RAMBAH/RAMBAL).

The SRAM control registers occupy eight bytes of address space, but not all locations are implemented. Unimplemented bits are read as zeros, and writes have no effect. Refer to **APPENDIX D REGISTER SUMMARY** for register block address map and register bit/field definitions.

6.2 SRAM Array Address Mapping

Base address registers RAMBAH and RAMBAL are used to specify the SRAM array base address in the memory map. RAMBAH and RAMBAL can only be written while the SRAM is in low-power stop mode (RAMMCR STOP = 1) and the base address lock is disabled (RAMMCR RLCK = 0). RLCK can be written once only to a value of one; subsequent writes are ignored. This prevents accidental remapping of the array.

NOTE

Because ADDR[23:20] always follow the state of ADDR19 on CPU16-based MCUs, the ADDR[23:20] bits in the SRAM base address high register (RAMBAH) must be programmed to the same state as the ADDR19 bit in order for the array to be accessible. If the array is mapped between IMB addresses $080000 and $F7FFFF, it will remain inaccessible until reset occurs, or it is remapped outside of this range. Refer to **3.5 CPU16 Memory Mapping** for more information.

6.3 SRAM Array Address Space Type

The RASP[1:0] bits in RAMMCR determine the SRAM array address space type. RASP0 determines whether the SRAM module responds to both program and data space accesses or to program space accesses only.

RASP1 specifies whether accesses to the SRAM module can be made in supervisor mode only, or in either supervisor or user mode. If supervisor-only access is selected, accesses in user mode are ignored by the SRAM control logic and can be decoded externally. Because the CPU16 operates in supervisor mode only, RASP1 has no effect. **Table 6-1** shows RASP[1:0] encodings.

Table 6-1 SRAM Array Address Space Type

RASP[1:0]	Space
X0	Program and data accesses
X1	Program access only

Refer to **5.5.1.7 Function Codes** for more information concerning address space types and program/data space access.

6.4 Standby Power Loss Detection

The power down status (PDS) bit in RAMMCR monitors the V_{STBY} standby power input during normal operation and while V_{DD} is powered down. When written to one, PDS will remain set, even if V_{DD} is powered down, so long as the minimum specified RAM standby voltage is maintained on the V_{STBY} pin. Should V_{STBY} fall below the specified minimum, PDS will be cleared to zero and will remain so until again written to one. Software can also write PDS to zero if it is necessary at some point to disable V_{STBY} monitoring.

6.5 Normal Access

The array can be accessed in byte, word, or long word fashion. A byte or aligned word access takes one bus cycle or two system clocks. A long word or misaligned word access requires two bus cycles. Refer to **5.6 Bus Operation** for more information concerning access times.

6.6 Standby and Low-Power Stop Operation

Standby and low-power modes should not be confused. Standby mode maintains the RAM array when the main MCU power supply is turned off. Low-power stop mode allows the CPU16 to control MCU power consumption by disabling unused modules.

Relative voltage levels of the MCU V_{DD} and V_{STBY} pins determine whether the SRAM is in standby mode. SRAM circuitry switches to the standby power source when V_{DD} drops below specified limits. If the specified standby supply voltage level is maintained during the transition, no memory loss will occur during switching. The RAM array cannot be accessed while the SRAM module is powered by V_{STBY}. If standby operation is not desired, connect the V_{STBY} pin to V_{SS}.

I_{SB} (SRAM standby current) values may vary while V_{DD} transitions occur. Refer to APPENDIX A ELECTRICAL CHARACTERISTICS for standby switching and power consumption specifications.

6.7 Reset

Reset places the SRAM in low-power stop mode, enables program space access, and clears the base address registers and the register lock bit. These actions make it possible to write a new base address into the RAMBAH and RAMBAL registers.

When a synchronous reset occurs while a byte or word SRAM access is in progress, the access is completed. If reset occurs during the first word access of a long-word operation, only the first word access is completed. If reset occurs during the second word access of a long-word operation, the entire access is completed. Data being read from or written to the RAM may be corrupted by an asynchronous reset. Refer to **5.7 Reset** for more information.

SECTION 7
MASKED ROM MODULE

Masked ROM modules (MRMs) are present only on the MC68HC16R1 and the MC68HC16R3 devices. Each MRM consists of a fixed-location control register block and a mask-programmed ROM array that can be mapped to a user-specified base address. The contents of the MRM control registers, like the array itself, are typically programmed with user-specified default values. Data can be read from the masked ROM array in byte, word, or long word fashion.

The MC68HC16R1 has a single 48-Kbyte MRM. The MC68HC16R3 has one 32-Kbyte MRM and one 64-Kbyte MRM.

7.1 MRM Register Block

There are three MRM control registers: the masked ROM module configuration register (MRxMCR) and the ROM array base address registers (ROMxBAH and ROMxBAL). In addition, the MRM register block contains signature registers (RxSIGHI and RxSIGLO), and ROM bootstrap words (ROMxBS[0:3]).

On the MC68HC16R1, the control register block for the 48-Kbyte masked ROM module starts at location $YFF820. On the MC68HC16R3, the control register blocks for the 32-Kbyte and 64-Kbyte masked ROM modules start at locations $YFF820 and $YFF840, respectively. The following register descriptions apply to the corresponding register in any of the control blocks. MRxMCR, for example, refers to MR1MCR on both the MC68HC16R1 and the MC68HC16R3 and MR2MCR on the MC68HC16R3.

The MRM control registers occupy 32 bytes of address space, but not all locations are implemented. Unimplemented bits are read as zeros, and writes have no effect. Refer to **APPENDIX D REGISTER SUMMARY** for register block address map and register bit/field definitions.

7.2 MRM Array Address Mapping

Base address registers ROMxBAH and ROMxBAL are used to specify the ROM array base address in the memory map. Although the base address contained in ROMxBAH and ROMxBAL is mask-programmed, these registers can be written after reset to change the default array address if the base address lock bit (LOCK in MRxMCR) is not masked to a value of one.

Masked ROM arrays sized to an integral power of two must be mapped to a base address that is an integer multiple of the array size. Thus, a 32-Kbyte array must be mapped to a base address that is an integer multiple of $8000, and a 64-Kbyte array must be mapped to a base address that is an integer multiple of $10000.

The 48-Kbyte MRM, like the 64-Kbyte MRM, must be mapped to a base address that is an integer multiple of $10000. The array only occupies the first 48-Kbytes of the 64-Kbyte block of address space into which it is mapped. Other memory resources (subject to block size and base address constraints) can be mapped immediately following the 48-Kbyte array and will remain accessible.

A masked ROM module should not be mapped to any address that overlaps other MCU control registers or memory arrays. Overlap will result in array and/or register inaccessibility and must be avoided.

ROMxBAH and ROMxBAL can only be written while the MRM is in low-power stop mode (MRxMCR STOP = 1) and the base address lock is disabled (MRxMCR LOCK = 0). LOCK can be written once only to a value of one. This prevents accidental remapping of the array.

7.3 MRM Array Address Space Type

The ASPC[1:0] bits in MRxMCR determine the MRM array address space type. ASPC0 determines whether the masked ROM module responds to both program and data space accesses or to program space accesses only.

ASPC1 specifies whether accesses to the MRM can be made in supervisor mode only, or in either supervisor or user mode. If supervisor-only access is selected, accesses in user mode are ignored by the MRM control logic and can be decoded externally. Because the CPU16 operates in supervisor mode only, ASPC1 has no effect.

The default value of ASPC[1:0] is established during mask programming but can be changed after reset if the LOCK bit in the MRxMCR has not been masked to a value of one.

Table 7-1 shows ASPC[1:0] field encodings.

Table 7-1 ROM Array Space Field

ASPC[1:0]	State Specified
X0	Program and data accesses
X1	Program access only

Refer to **5.5.1.7 Function Codes** for more information concerning address space types and program/data space access.

7.4 Normal Access

The array can be accessed in byte, word, or long word fashion. A byte or aligned word access takes one bus cycle or two system clocks. A long word or misaligned word access requires two bus cycles.

To accomodate migration from slower development memories, the MRM can insert wait states into each array-access bus cycle.The number of wait states inserted is determined by the value of WAIT[1:0] in MRxMCR. Two, three, four, or five system clock cycle accesses can be specified. The default value of WAIT[1:0] is established during mask programming but can be changed after reset if the LOCK bit in the MRxMCR has not been masked to a value of one.

Table 7-2 shows WAIT[1:0] field encodings.

Table 7-2 Wait States Field

WAIT[1:0]	Number of Wait States	Clocks per Bus Cycle
00	0	3
01	1	4
10	2	5
11	–1	2

Refer to **5.6 Bus Operation** for more information concerning access times.

7.5 Low-Power Stop Mode Operation

Low-power stop mode minimizes module power consumption. Setting the STOP bit in MRxMCR places the MRM in low-power stop mode. In low-power stop mode, the array cannot be accessed. On the MC68HC16R1 and the MC68HC16R3, the reset state of STOP for the 32-, 48-, and 64-Kbyte MRMs is the complement of the logic state of DATA14 during reset. The MRM exits low-power stop mode when the STOP bit is cleared.

7.6 ROM Signature

Signature registers RxSIGHI and RxSIGLO contain a user-specified mask-programmed signature pattern. A user-written signature algorithm allows the user to verify ROM array content. Once masked, the contents of RxSIGHI and RxSIGLO cannot be changed.

7.7 Reset

The behavior of the MRM following reset is determined by the default values programmed into the MRxMCR $\overline{\text{BOOT}}$, LOCK, ASPC[1:0], and WAIT[1:0] bits. The default array base address is determined by the values programmed into ROMxBAL and ROMxBAH.

When the mask programmed value of the MRxMCR $\overline{\text{BOOT}}$ bit is zero, the contents of MRM bootstrap words ROMxBS[0:3] are used as reset vectors. When $\overline{\text{BOOT}}$ is masked to one, the MRM will not respond to reset vector fetches, and another MRM or an external device should be configured to support bootstrap operation. Refer to **5.9.4 Chip-Select Reset Operation** for more information concerning external boot ROM selection.

SECTION 8
FLASH EEPROM MODULE

The flash EEPROM modules serve as non-volatile, fast-access, electrically erasable and programmable read-only memory. These modules are present only in the MC68HC916R1 and the MC68HC916R3.

The MC68HC916R1 has one 16-Kbyte flash module and one 32-Kbyte flash module. The MC68HC916R3 has three 32-Kbyte flash modules.

Each flash module consists of a fixed location control register block with user-programmable default values and a reprogrammable flash EEPROM array that can be mapped to a user-specified address. Data can be read from the flash array in byte, word, or long word fashion.

The flash array and the shadow bits that provide default values for the flash control registers are programmable and erasable under software control. The program/erase voltage is supplied via external V_{FPE} pins. Data is programmed in byte or aligned word fashion. Simultaneous programming of multiple bytes or words is not supported. The flash EEPROM modules support bulk erase only, and have a program/erase life of at least 100 cycles.

The flash EEPROM modules have hardware interlocks which protect stored data from corruption by accidental enabling of the program/erase voltage to the flash EEPROM arrays.The hardware interlocks make inadvertent programming or erasure highly unlikely.

8.1 Flash EEPROM Register Block

There are five flash EEPROM control registers: the flash EEPROM module configuration register (FEExMCR), the flash EEPROM test register (FEExTST), the flash EEPROM base address registers (FEExBAH and FEExBAL), and the flash EEPROM control register (FEExCTL). Four additional flash EEPROM words in the control register block (FEExBS[0:3]) can contain bootstrap information for use during reset. The flash EEPROM control registers are located in supervisor data space. Refer to **APPENDIX D REGISTER SUMMARY** for register and bit field information.

On the MC68HC916R1, the control register blocks for the 16- and 32-Kbyte flash EEPROM modules start at locations $YFF800 and $YFF820, respectively. On the MC68HC916R3, the control register blocks for the three 32-Kbyte flash EEPROM modules start at locations $YFF800, $YFF820, and $YFF840, respectively. The following register descriptions apply to the corresponding register in any of the control blocks. FEExMCR, for example, refers to FEE1MCR and FEE2MCR on the MC68HC916R1 and FEE1MCR, FEE2MCR, and FEE3MCR on the MC68HC916R3.

A number of control register bits have associated shadow bits that allow the flash control registers to take on user-specified values at reset. Shadow bits are programmed in the same manner as any location in the flash array. Data is programmed into the shadow bits simply by using the address of the control register as the address to be programmed. The contents of a register with shadow bits are not updated immediately

after programming but change only after the next reset occurs. When a flash array is erased, its associated shadow bits are also erased.

8.2 Flash EEPROM Array Address Mapping

Base address registers FEExBAH and FEExBAL are used to specify the flash array base address in the memory map. Shadow bits in FEExBAH and FEExBAL are usually programmed once to specify a default base address for the flash array. Although it is usually unnecessary to change the array base address, a new value can be written to FEExBAH and FEExBAL (as opposed to reprogramming the FEExBAH and FEExBAL shadow bits which can be done at any time) if the LOCK bit in FEExMCR has not been programmed to one. Writes to FEExBAH and FEExBAL will not change the states of the shadow bits associated with these registers. Shadow bits must be changed using the flash programming algorithm.

Flash arrays sized to an integral power of two must be mapped to a base address that is an integer multiple of the array size. Thus, a 16-Kbyte array must be mapped to a base address that is an integer multiple of $4000, and a 32-Kbyte array must be mapped to a base address that is an integer multiple of $8000. The flash arrays can be mapped to create contiguous blocks of address space, subject to block size and base address constraints.

A flash EEPROM module should not be mapped to any address that overlaps other MCU control registers or memory arrays. Overlap will result in array and/or register inaccessibility and must be avoided.

8.3 Flash EEPROM Array Address Space Type

The ASPC[1:0] bits in FEExMCR determine the flash array address space type. ASPC0 determines whether the flash array responds to both program and data space accesses or to program space accesses only.

ASPC1 specifies whether accesses to the flash array can be made in supervisor mode only, or in either supervisor or user mode. If supervisor-only access is selected, accesses in user mode are ignored by the flash EEPROM module control logic and can be decoded externally. Because the CPU16 operates in supervisor mode only, ASPC1 has no effect.

8.4 Normal Access

The flash EEPROM array can be accessed in byte, word, or long word fashion. A byte or aligned word access takes one bus cycle which is typically two system clocks. A long word or misaligned word access requires two bus cycles.

To accommodate migration from slower development memories, the flash EEPROM module can insert wait states into each array access bus cycle. The number is wait states inserted is determined by the value of WAIT[1:0] in FEExMCR. Two, three, four, or five system clock cycle accesses can be specified. The default value of WAIT[1:0] is obtained from associated flash shadow bits but can be changed after reset if the LOCK bit in FEExMCR has been programmed to zero.

Table 8-1 shows WAIT[1:0] field encodings.

Table 8-1 Wait States Field

WAIT[1:0]	Number of Wait States	Clocks per Bus Cycle
00	0	3
01	1	4
10	2	5
11	–1	2

Refer to **5.6 Bus Operation** for more information concerning access times.

8.5 Low-Power Stop Mode Operation

Low-power stop mode minimizes module power consumption. Setting the STOP bit in FEExMCR places the flash EEPROM module in low-power stop mode. In low-power stop mode, the flash EEPROM array cannot be accessed. After reset, STOP assumes the value of the boolean expression STOP = $\overline{\text{DATA14}}$ + SB, where SB is the STOP shadow bit. The flash EEPROM module exits low-power stop mode when the STOP bit is cleared.

8.6 Reset

The behavior of the flash EEPROM module following reset is determined by the default values programmed into the FEExMCR $\overline{\text{BOOT}}$, LOCK, ASPC[1:0], and WAIT[1:0] shadow bits. The default array base address is determined by the values programmed into the FEExBAH and FEExBAL shadow bits.

When the value of the $\overline{\text{BOOT}}$ shadow bit is zero, the contents of the flash EEPROM module bootstrap words (FEExBS[0:3]) are used as reset vectors. When the $\overline{\text{BOOT}}$ shadow bit is programmed to one, the flash module will not respond to reset vector fetches, and another flash module or an external device should be configured to support bootstrap operation.

8.7 Program/Erase Operation

An erased flash bit has a logic state of one. A bit must be programmed to change its state from one to zero. Erasing a bit returns it to a logic state of one. Programming and erasing the flash requires a series of control register and array address writes. The same procedure is used to program addresses in the flash array and control registers that contain flash shadow bits. Programming is restricted to a single byte or aligned word at a time. The entire array and the shadow register bits are erased at the same time.

When multiple flash modules share a single V_{FPE} pin, program or erase only one module at a time. Simultaneous programming of multiple modules is not supported and will violate the V_{FPE} current specifications (I_{FPE}) outlined in APPENDIX A ELECTRICAL CHARACTERISTICS**.** Normal accesses to modules that are not being programmed or

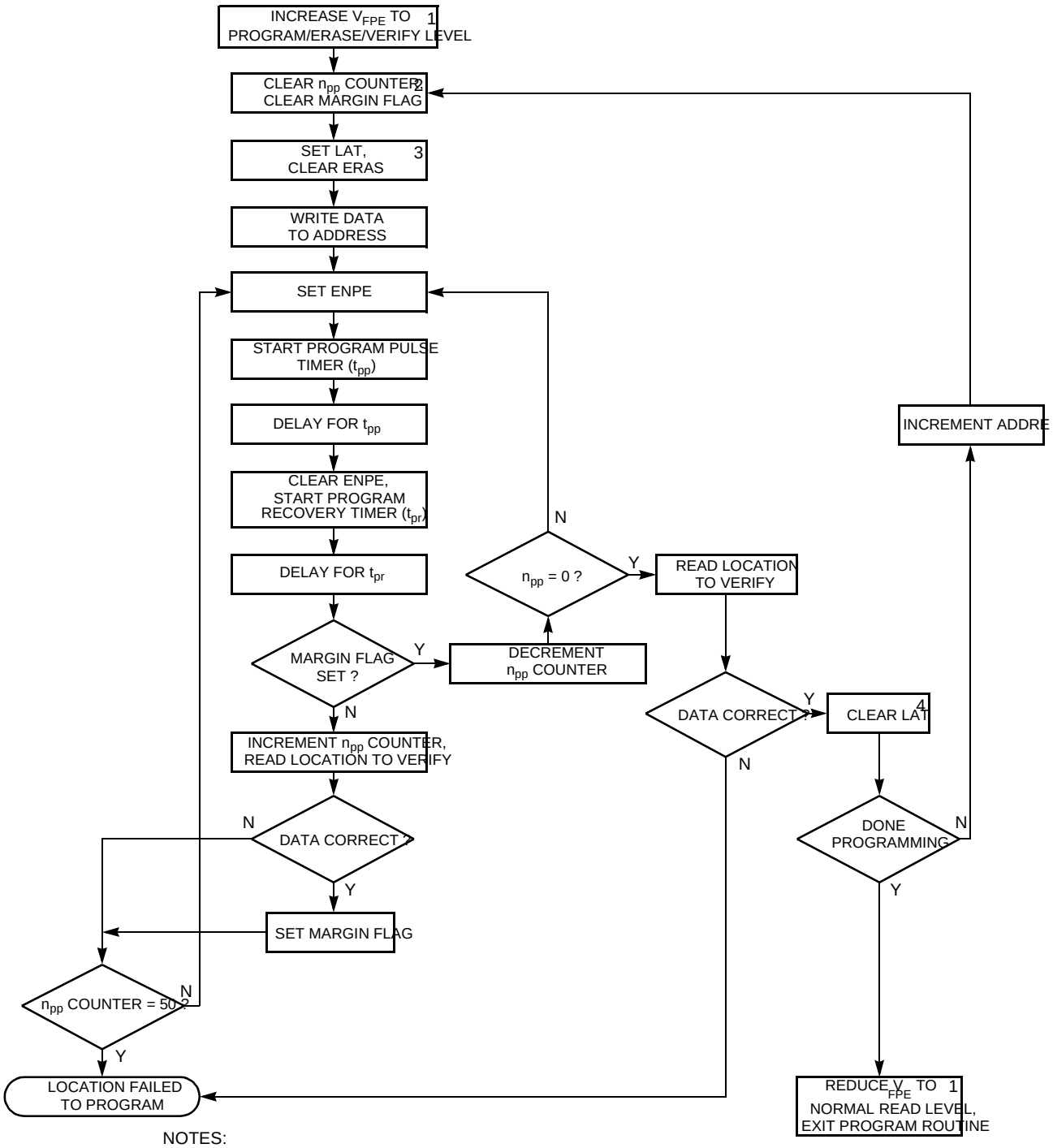
erased are not affected by programming or erasure of another flash module, even when these modules share a single V_{FPE} pin.

The following paragraphs give step-by-step procedures for programming and erasing the flash EEPROM array. Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for information on program and erase specifications for the flash EEPROM module.

8.7.1 Programming Sequence

Use the following steps to program a flash EEPROM array location or the shadow bits in a flash control register. The flowchart in **Figure 8-1** outlines this procedure. Refer to **8.7.3 Program/Erase Voltage Signal Conditioning** for information on maintaining the proper relationship between V_{FPE} and V_{DD} during flash programming.

1. Raise the voltage applied to the to the V_{FPE} pin from V_{DD} (the normal read level) to the program/erase level. Accurate V_{FPE} control is best accomplished with a dedicated flash program/erase voltage generator integrated circuit.
2. Set the LAT bit in FEExCTL; leave all other FEExCTL bits clear. This enables the programming address and data latches.
3. Write the desired data to the address to be programmed. This latches the address to be programmed and the programming data.
4. Set both the LAT and ENPE bits in FEExCTL. This starts the programming pulse.
5. Allow the programming pulse time to elapse. The programming pulse time is specified by the t_{pp} parameter in APPENDIX A ELECTRICAL CHARACTERISTICS**.**
6. Leaving LAT set, clear ENPE in FEExCTL. This stops the programming pulse.
7. Allow the program recovery time delay to elapse while the high voltage to the array is shut off. The program recovery time is specified by the parameter t_{pr} in **APPENDIX A ELECTRICAL CHARACTERISTICS.**
8. Read the address to verify that it has been programmed. When reading a flash control register with shadow bits, the LAT bit in FEExCTL must be set or the read will return the contents of the control register and not the state of its shadow bits.
9. If the data read is not the same as that programmed, repeat steps 4 through 7 until the location is programmed or until the maximum number of programming pulses has been applied. The maximum number of programming pulses is specified by the parameter n_{pp} in APPENDIX A ELECTRICAL CHARACTERISTICS**.**
10. If the data read is the same as that programmed, re-apply the same number of programming pulses as first issued in steps 4 through 7. This provides 100% programming margin to the address just programmed.
11. After issuing the margin pulses, re-read the address just programmed to verify that it remains programmed.
12. Clear the LAT bit in FEExCTL. This disables the programming address and data latches.
13. If more locations are to be programmed, repeat steps 2 through 12.
14. Lower the voltage applied to the V_{FPE} pin from the program/erase level to V_{DD} (the normal read level).



NOTES:

1. SEE APPENDIX A ELECTRICAL CHARACTERISTICS FOR V_{FPE} PIN VOLTAGE SEQUENCING.
2. THE MARGIN FLAG IS A SOFTWARE-DEFINED FLAG THAT INDICATES WHETHER THE PROGRAM SEQU GENERATING PROGRAM PULSES OR MARGIN PULSES.
3. TO SIMPLIFY THE PROGRAM OPERATION, THE V_{FPE} BIT IN FEExCTL CAN BE SET.
4. CLEAR V_{FPE} BIT ALSO IF ROUTINE USES THIS FUNCTION.

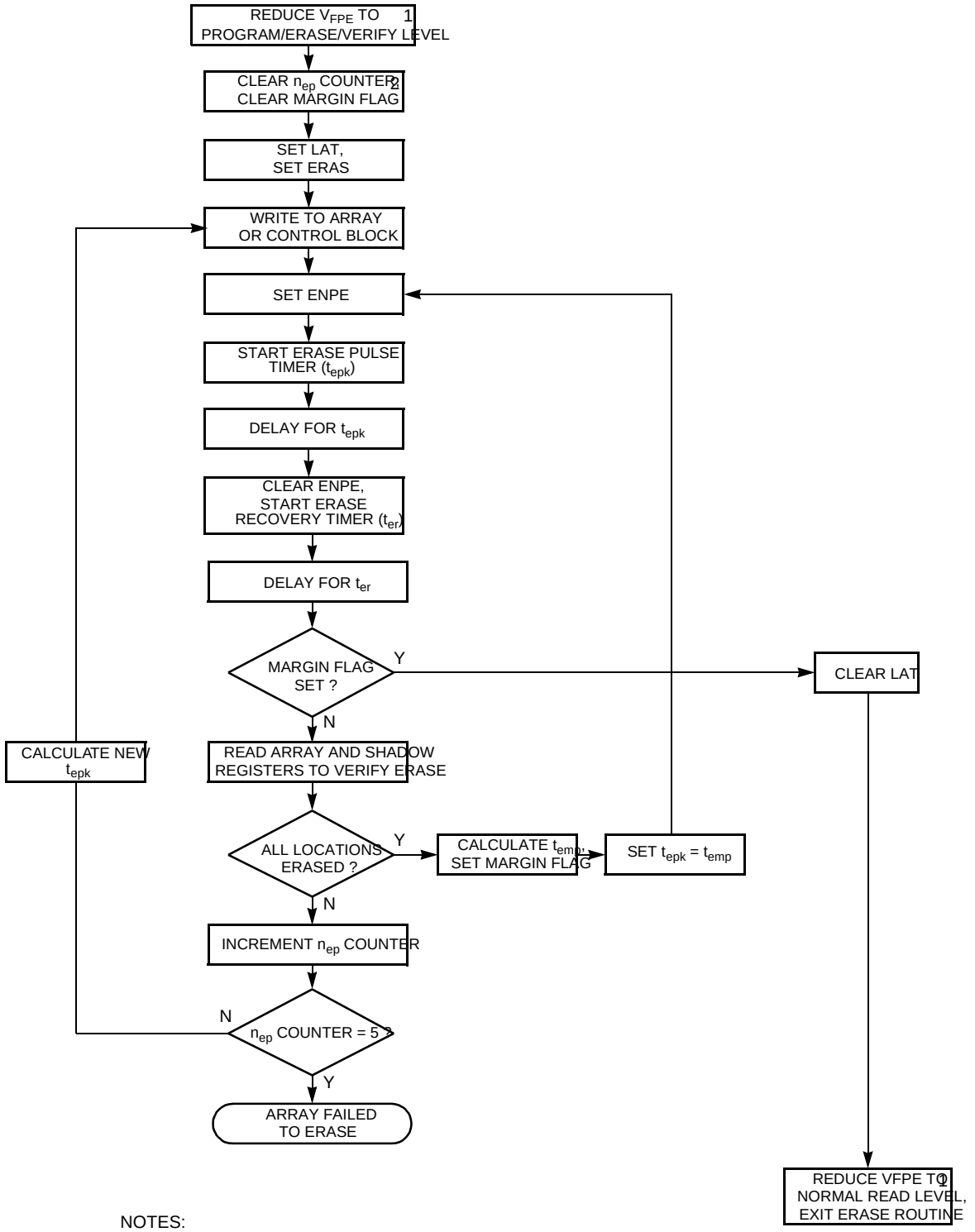
FEEPROM PGM FLOW1 USM

Figure 8-1 Programming Flowchart

8.7.2 Erasure Sequence

Use the following steps to erase a flash EEPROM array. The flowchart in **Figure 8-2** outlines this procedure. Refer to **8.7.3 Program/Erase Voltage Signal Conditioning** for information on maintaining the proper relationship between V_{FPE} and V_{DD} during flash erasure.

1. Raise the voltage applied to the to the V_{FPE} pin from V_{DD} (the normal read level) to the program/erase level. Accurate V_{FPE} control is best accomplished with a dedicated flash program/erase voltage generator integrated circuit.
2. Set the LAT and ERAS bits in FEExCTL; leave all other FEExCTL bits clear. This configures the module for erasure.
3. Perform a write to any flash array location or flash control register. The data written does not matter as this dummy write is part of the interlock sequence required for erasure.
4. Set the LAT, ERAS, and ENPE bits in FEExCTL. This starts the erase pulse.
5. Allow the erase pulse time to elapse. The erase pulse time is specified by the t_{epk} parameter in **APPENDIX A ELECTRICAL CHARACTERISTICS.**
6. Keeping LAT and ERAS set, clear ENPE in FEExCTL. This stops the erase pulse.
7. Allow the erase recovery time delay to elapse while the high voltage to the array is shut off. The program recovery time is specified by the parameter t_{er} in **APPENDIX A ELECTRICAL CHARACTERISTICS.**
8. Read the entire flash array and its control registers to verify erasure. The erased state of a flash array word is $FFFF. The erased states of control registers with shadow bits are listed in **APPENDIX D REGISTER SUMMARY.** When reading a flash control register with shadow bits, the LAT bit in FEExCTL must be set or the read will return the contents of the control register and not the state of its shadow bits.
9. If all array locations and control registers with shadow bits are not erased, calculate the new erase pulse time as $t_{epk} = t_{ei}$ (erase pulse increment) $\times$ erase pulse number. Repeat steps 3 through 8 until all locations erase or until the maximum number of erases pulses has been applied. The maximum number of erase pulses is specified by the parameter n_{ep} in APPENDIX A ELECTRICAL CHARACTERISTICS**.**
10. If all locations are erased, calculate the erase margin pulse time, t_{epm} (the summation of all previous erase pulses) and repeat steps 3 through 8 for the single margin pulse. This applies 100% erase margin to the array.
11. Clear the LAT and ERAS bits in FEExCTL. This permits normal access to the array.
12. Lower the voltage applied to the V_{FPE} pin from the program/erase level to V_{DD} (the normal read level).



NOTES:

1. SEE APPENDIX A ELECTRICAL CHARACTERISTICS FOR V_{FPE} PIN VOLTAGE SEQUENCING.
2. THE MARGIN FLAG IS A SOFTWARE-DEFINED FLAG THAT INDICATES WHETHER THE PROGRAM SEQUENCE IS GENERATING ERASE PULSES OR MARGIN PULSES.

Figure 8-2 Erasure Flowchart

8.7.3 Program/Erase Voltage Signal Conditioning

A voltage of at least V_{DD} - 10% must be applied to the V_{FPE} pins at all times, or damage to the flash EEPROM module can occur as a result of power-up and power-down transients. V_{FPE} also must not rise to the program/erase level before V_{DD} reaches its minimum operating level. **Figure 8-3** shows the required operating envelope for V_{DD} and V_{FPE}. Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for V_{FPE} maximum and operating specifications.

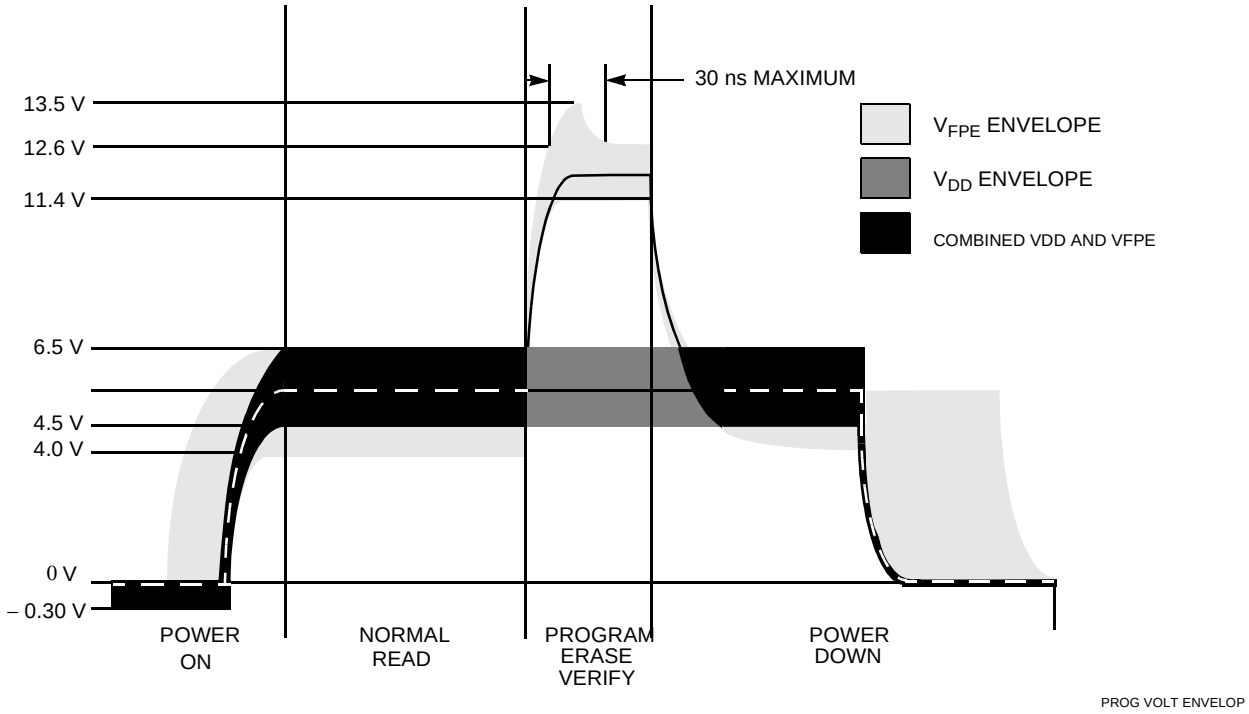


Figure 8-3 Programming Voltage Envelope

Use of a dedicated circuit to generate the 12.0V ± 5% required for program and erase operations is recommended. In addition, the V_{FPE} conditioning/protection circuit shown in **Figure 8-4** should be used to maintain proper voltage levels on the V_{FPE} pin and to protect it from power-up and power-down transients.

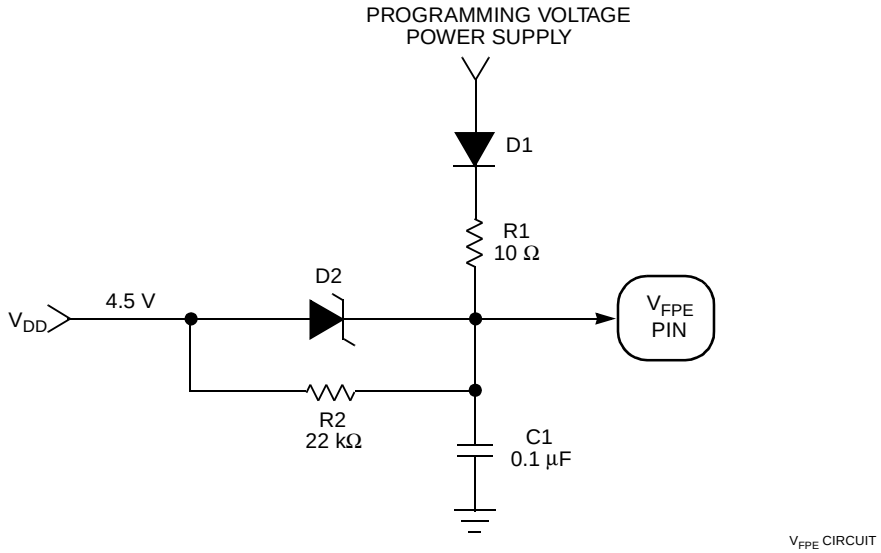


Figure 8-4 V_{FPE} Conditioning Circuit

V_{FPE} is pulled up to V_{DD} via Schottky diode D2. Application of the programming voltage via diode D1 reverse-biases D2, protecting V_{DD} from excessive reverse current. D2 also protects the flash module from damage should the programming voltage fall to zero. The programming voltage power supply should be adjusted to compensate for the forward-bias drop across D1. The charge time constant of R1 and C1 filters transients, while R2 provides a discharge bleed path for C1. Allow for RC charge and discharge times when applying and removing system power. When using this circuit, keep leakage from external devices connected to the V_{FPE} pin low to minimize diode voltage drop.

SECTION 9
BLOCK-ERASABLE FLASH EEPROM

The 2-Kbyte block-erasable flash EEPROM module (BEFLASH) serves as non-volatile, fast access, electrically erasable and programmable read-only memory. Unlike the standard flash EEPROM module which is bulk-erasable only, the BEFLASH is split into eight individually erasable sectors. This makes the BEFLASH particularly useful for storing parameter data that changes independently of the application's firmware. The BEFLASH also supports bootstrap operation and can be used for normal program storage.

The BEFLASH is present only on the MC68HC916R1 and the MC68HC916R3.

Each BEFLASH module consists of a fixed location control register block with user-programmable default values and a reprogrammable flash EEPROM array with sector selectable erase capability that can be mapped to a user-specified address. Data can be read from the flash array in byte, word, or long word fashion.

The block-erasable flash array and the shadow bits that provide default values for the BEFLASH control registers are programmable and erasable under software control. The program/erase voltage is supplied via an external V_{FPE} pin. Data is programmed in byte or aligned word fashion. Simultaneous programming of multiple bytes or words is not supported. As shown in **Table 9-1**, eight asymmetrically-sized sectors and a block of control register shadow bits comprise the BEFLASH array. Each sector can be erased independently of the other sectors, or the entire BEFLASH (including control register shadow bits) can be bulk erased. The BEFLASH array has a program/erase life of at least 100 cycles.

Table 9-1 Block-Erasable Flash Sectoring

Block	Offset From Array Base Address	Size (Bytes)
0	$0000 - $007F	128
1	$0080 - $00FF	128
2	$0100 - $017F	128
3	$0180 - $01FF	128
4	$0200 - $02FF	256
5	$0300 - $03FF	256
6	$0400 - $05FF	512
7	$0600 - $07FF	512
Control Registers	$YFF7A0 - $YFF7BF	32

The BEFLASH module has hardware interlocks which protect stored data from corruption by accidental enabling of the program/erase voltage to the block-erasable flash EEPROM array. The hardware interlocks make inadvertent programming or erasure highly unlikely.

9.1 Block-Erasable Flash EEPROM Register Block

There are five block-erasable flash EEPROM control registers: the block-erasable flash EEPROM module configuration register (BFEMCR), the block-erasable flash EEPROM test register (BFETST), the block-erasable flash EEPROM base address registers (BFEBAH and BFEBAL), and the block-erasable flash EEPROM control register (BFECTL). Four additional flash EEPROM words in the control register block (BFEBS[0:3]) can contain bootstrap information for use during reset. The block-erasable flash EEPROM control registers are located in supervisor data space. On both the MC68HC916R1 and the MC68HC916R3, the 32-byte BEFLASH control register block begins at address $YFF7A0. Refer to **APPENDIX D REGISTER SUMMARY** for register and bit field information.

A number of control register bits have associated shadow bits that allow the block-erasable flash control registers to take on user-specified values at reset. Shadow bits are programmed in the same manner as any location in the BEFLASH array. Data is programmed into the shadow bits simply by using the address of the control register as the address to be programmed. The contents of a register with shadow bits are not updated immediately after programming but change only after the next reset occurs. The BEFLASH control register shadow bits are erased only when module is bulk erased.

9.2 Block-Erasable Flash EEPROM Array Address Mapping

Base address registers BFEBAH and BFEBAL are used to specify the block-erasable flash array base address in the memory map. Shadow bits in BFEBAH and BFEBAL are usually programmed once to specify a default base address for the flash array. Although it is usually unnecessary to change the array base address, a new value can be written to BFEBAH and BFEBAL (as opposed to reprogramming the BFEBAH and BFEBAL shadow bits which can be done at any time) if the LOCK bit in BFEMCR has not been programmed to one. Writes to BFEBAH and BFEBAL will not change the states of the shadow bits associated with these registers. Shadow bits must be changed using the flash programming algorithm.

The 2-Kbyte BEFLASH array must be mapped to a base address that is an integer multiple of $0800 (the 2048-byte array size). The BEFLASH array can be mapped preceding or following other flash arrays to create contiguous blocks of address space, subject to block size and base address constraints.

The BEFLASH module should not be mapped to any address that overlaps other MCU control registers or memory arrays. Overlap will result in array and/or register inaccessibility and must be avoided.

9.3 Block-Erasable Flash EEPROM Array Address Space Type

The ASPC[1:0] bits in BFEMCR determine the block-erasable flash array address space type. ASPC0 determines whether the BEFLASH array responds to both program and data space accesses or to program space accesses only.

ASPC1 specifies whether accesses to the BEFLASH array can be made in supervisor mode only, or in either supervisor or user mode. If supervisor-only access is selected, accesses in user mode are ignored by the BEFLASH module control logic and can be decoded externally. Because the CPU16 operates in supervisor mode only, ASPC1 has no effect.

9.4 Normal Access

The block-erasable flash EEPROM array can be accessed in byte, word, or long word fashion. Unlike the standard bulk-erasable flash EEPROM modules, the BEFLASH module does not support programmable wait state insertion. All byte and aligned word accesses take one bus cycle which is two system clocks. All long word or misaligned word accesses require two bus cycles. Refer to **5.6 Bus Operation** for more information concerning access times.

9.5 Low-Power Stop Mode Operation

Low-power stop mode minimizes module power consumption. Setting the STOP bit in BFEMCR places the block-erasable flash EEPROM module in low-power stop mode. In low-power stop mode, the BEFLASH array cannot be accessed. After reset, STOP assumes the value of the boolean expression STOP = $\overline{\text{DATA15}}$ + SB, where SB is the STOP shadow bit. The BEFLASH module exits low-power stop mode when the STOP bit is cleared.

9.6 Reset

The behavior of the block-erasable flash EEPROM module following reset is determined by the default values programmed into the BFEMCR $\overline{\text{BOOT}}$, LOCK, and ASPC[1:0] shadow bits. The default array base address is determined by the values programmed into the BFEBAH and BFEBAL shadow bits.

When the value of the $\overline{\text{BOOT}}$ shadow bit is zero, the contents of the block-erasable flash EEPROM module bootstrap words (BFEBS[0:3]) are used as reset vectors. When the $\overline{\text{BOOT}}$ shadow bit is programmed to one, the BEFLASH module will not respond to reset vector fetches, and another flash module or an external device should be configured to support bootstrap operation.

9.7 Program/Erase Operation

An erased flash bit has a logic state of one. A bit must be programmed to change its state from one to zero. Erasing a bit returns it to a logic state of one. Programming and erasing the block-erasable flash requires a series of control register and array address writes. The same procedure is used to program addresses in the BEFLASH array and control registers that contain flash shadow bits. Programming is restricted to a single byte or aligned word at a time. The entire array (including control register shadow bits) can be bulk erased, or any one of the eight sectors can be erased individually without affecting the other seven sectors or control register shadow bits.

When multiple flash modules share a single V_{FPE} pin, program or erase only one module at a time. Simultaneous programming of multiple modules is not supported and will

violate the V_{FPE} current specifications (I_{FPE}) outlined in APPENDIX A ELECTRICAL CHARACTERISTICS**.** Normal accesses to modules that are not being programmed or erased are not affected by programming or erasure of another flash module, even when these modules share a single V_{FPE} pin.

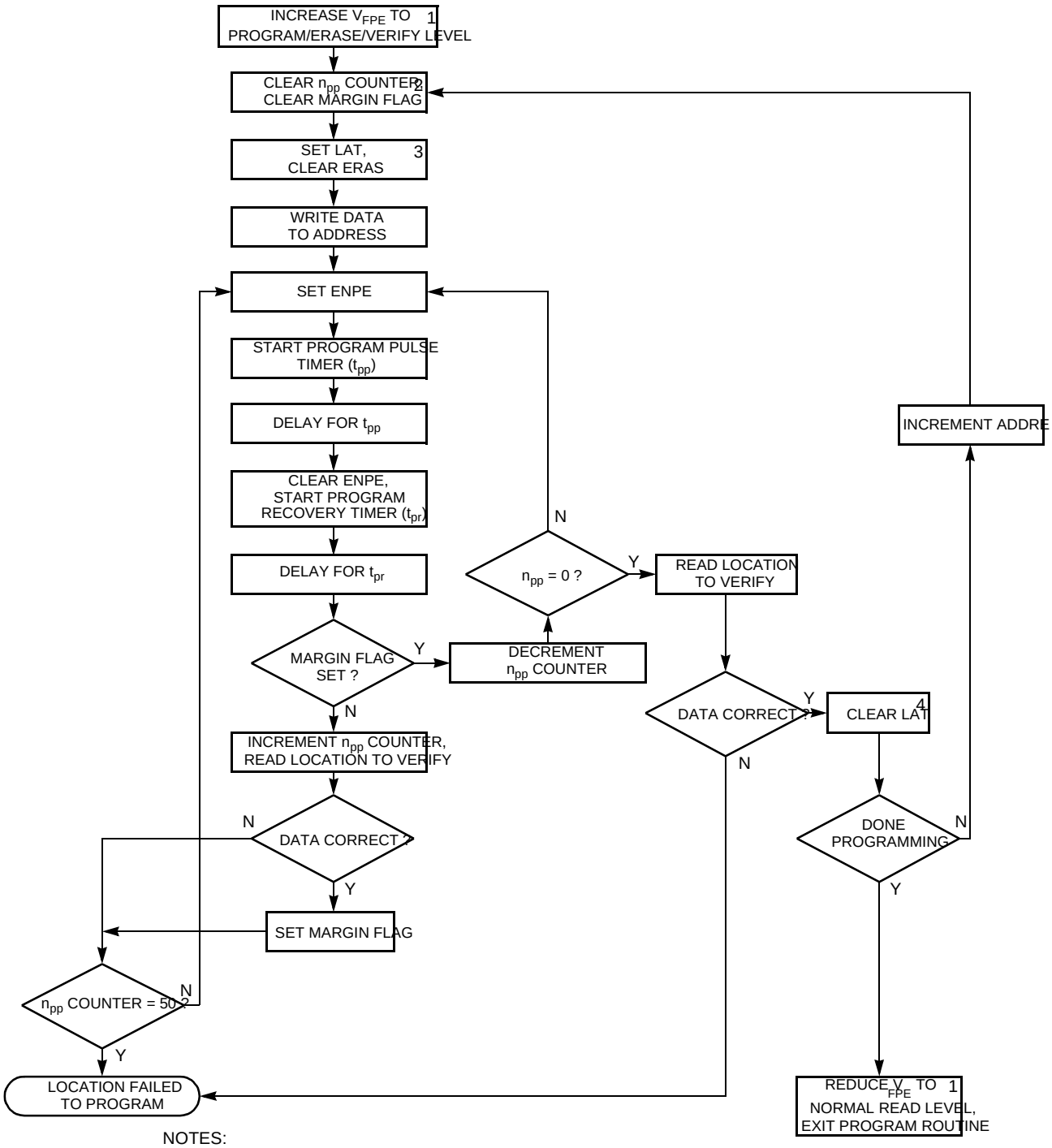
The following paragraphs give step-by-step procedures for programming and erasing the block-erasable flash EEPROM array. Refer to APPENDIX A ELECTRICAL CHARACTERISTICS**.** for information on program and erase specifications for the flash EEPROM module.

9.7.1 Programming Sequence

Use the following steps to program a BEFLASH array location or the shadow bits in a BEFLASH control register. The flowchart in **Figure 9-1** outlines this procedure. Refer to **9.7.3 Program/Erase Voltage Signal Conditioning** for information on maintaining the proper relationship between V_{FPE} and V_{DD} during flash programming.

1. Raise the voltage applied to the to the V_{FPE} pin from V_{DD} (the normal read level) to the program/erase level. Accurate V_{FPE} control is best accomplished with a dedicated flash program/erase voltage generator integrated circuit.
2. Set the LAT bit in BFECTL; leave all other BFECTL bits clear. This enables the programming address and data latches.
3. Write the desired data to the address to be programmed. This latches the address to be programmed and the programming data.
4. Set both the LAT and ENPE bits in BFECTL. This starts the programming pulse.
5. Allow the programming pulse time to elapse. The programming pulse time is specified by the t_{pp} parameter in APPENDIX A ELECTRICAL CHARACTERISTICS.
6. Leaving LAT set, clear ENPE in BFECTL. This stops the programming pulse.
7. Allow the program recovery time delay to elapse while the high voltage to the array is shut off. The program recovery time is specified by the parameter t_{pr} in **APPENDIX A ELECTRICAL CHARACTERISTICS.**
8. Read the address to verify that it has been programmed. When reading a flash control register with shadow bits, the LAT bit in BFECTL must be set or the read will return the contents of the control register and not the state of its shadow bits.
9. If the data read is not the same as that programmed, repeat steps 4 through 7 until the location is programmed or until the maximum number of programming pulses has been applied. The maximum number of programming pulses is specified by the parameter n_{pp} in APPENDIX A ELECTRICAL CHARACTERISTICS**.**
10. If the data read is the same as that programmed, re-apply the same number of programming pulses as first issued in steps 4 through 7. This provides 100% programming margin to the address just programmed.
11. After issuing the margin pulses, re-read the address just programmed to verify that it remains programmed.
12. Clear the LAT bit in BFECTL. This disables the programming address and data latches.
13. If more locations are to be programmed, repeat steps 2 through 12.
14. Lower the voltage applied to the V_{FPE} pin from the program/erase level to V_{DD}

(the normal read level).



NOTES:
1. SEE APPENDIX A ELECTRICAL CHARACTERISTICS FOR V_{FPE} PIN VOLTAGE SEQUENCING.
2. THE MARGIN FLAG IS A SOFTWARE-DEFINED FLAG THAT INDICATES WHETHER THE PROGRAM SEQU GENERATING PROGRAM PULSES OR MARGIN PULSES.
3. TO SIMPLIFY THE PROGRAM OPERATION, THE V_{FPE} BIT IN FEExCTL CAN BE SET.
4. CLEAR V_{FPE} BIT ALSO IF ROUTINE USES THIS FUNCTION.

BEFLASH PGM FLOW1 USM

Figure 9-1 Programming Flowchart

9.7.2 Erasure Sequence

Eight asymmetrically-sized sectors and a block of control register shadow bits comprise the BEFLASH array. Each sector can be erased independently of the other sectors, or the entire BEFLASH (including control register shadow bits) can be bulk erased. In the erase procedure that follows, the address chosen for the write in step 3 specifies whether a single BEFLASH sector or the entire BEFLASH array is erased. Use **Table 9-2** to determine the appropriate address for erasing one of the eight BEFLASH blocks or the entire array.

Table 9-2 BEFLASH Erase Operation Address Ranges

Block	Addresses Affected	Address Bits Used to Specify Block for Erasure						
		ADDR[23:11]	ADDR[10:6]	A5	A4	A3	A2	ADDR[1:0]
0	$0000 - $007F	BFEBAH:BFEBAL[1]	X[2]	1	0	0	0	X[2]
1	$0080 - $0100			1	0	0	1	
2	$0100 - $017F			1	0	1	0	
3	$0180 - $01FF			1	0	1	1	
4	$0200 - $02FF			1	1	0	0	
5	$0300 - $03FF			1	1	0	1	
6	$0400 - $05FF			1	1	1	0	
7	$0600 - $07FF			1	1	1	1	
Entire Array[3]	$0000 - $07FF $YFF7A0 - $YFF7BF			0	X	X	X	

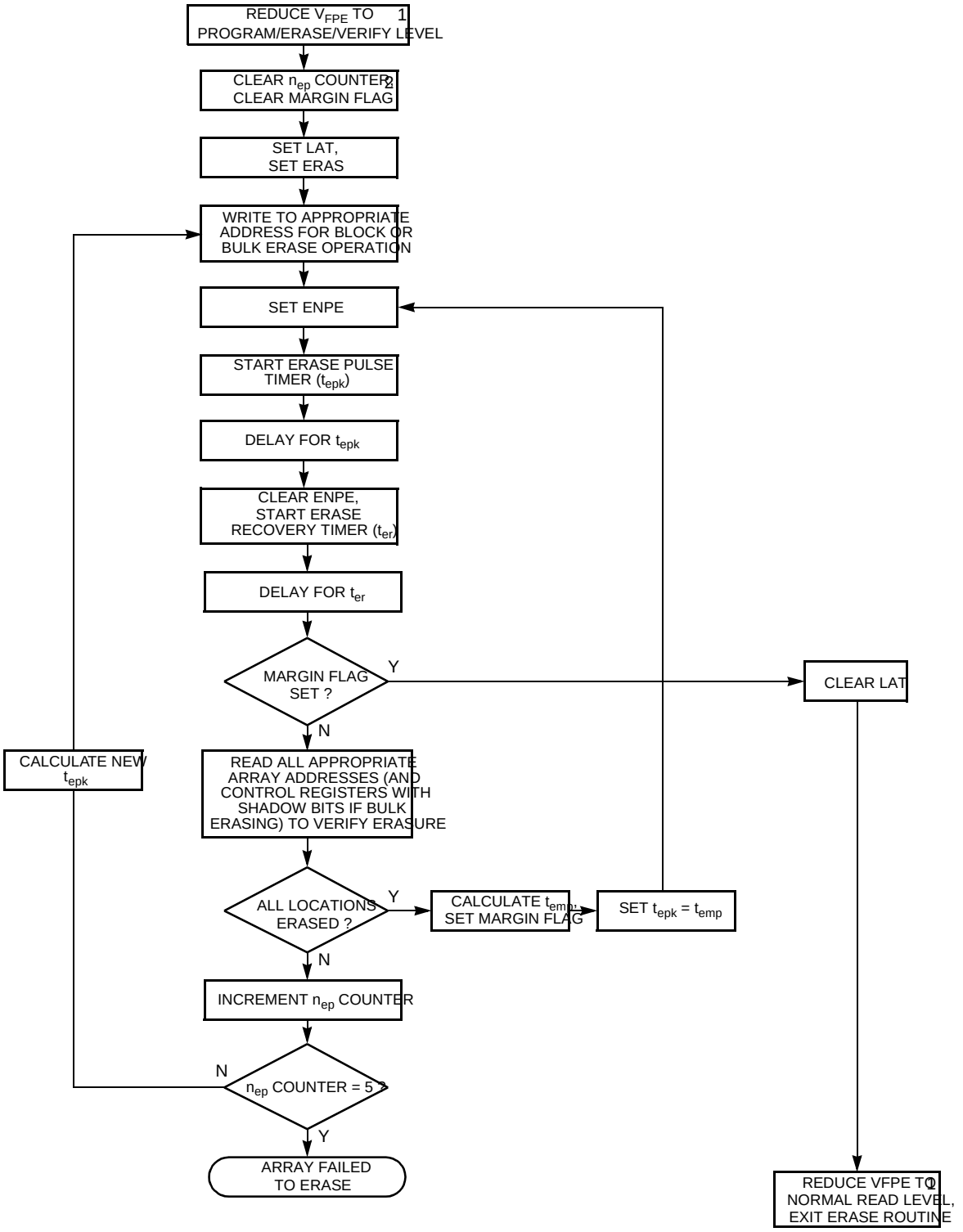
NOTES:
1. The block erasable flash base address high and low registers (BFEBAH and BFEBAL) specify ADDR[23:11] of the block to be erased.
2. These address bits are “don’t cares” when specifying the block to be erased.
3. Erasing the entire array also erases the BEFLASH control register shadow bits. In step 3 of the erase procedure that follows, writing to a control register will also bulk erase the BEFLASH array.

In order to erase block 0 of the BEFLASH array, **Table 9-2** states that the lower byte of the address specified in step 3 of the erase procedure must range from $20 to $23, $60 to $63, $A0 to $A3, or $E0 to $E3. Thus, if the BEFLASH array is mapped from $10000 to $107FF, valid addresses for this write would be $10x20, $10x21, $10x22, $10x23, $10x60, $10x61, $10x62, $10x63, $10xA0, $10xA1, $10xA2, $10xA3, $10xE0, $10xE1, $10xE2, and $10xE3, where $0 ð x ð $7.

Similarly, **Table 9-2** states in order to erase the entire BEFLASH array, the lower byte of the address specified in step 3 of the erase procedure must range from $00 to $1F, $40 to $5F, $80 to $9F, or $C0 to $DF. Thus, if the BEFLASH array is mapped from $10000 to $107FF, valid addresses for this write would be $10y0x, $10y1x, $10y4x, $10y5x, $10y8x, $10y9x, $10yCx, and $10yDx, where $0 ð y ð $7 and $0 ð x ð $F. In addition to these addresses, a write to a BEFLASH control register (i.e. BFEMCR) in step 3 of the erase procedure will also bulk erase the array.

Use the following steps to erase a block of or the entire BEFLASH array. The flowchart in **Figure 9-2** outlines this procedure. Refer to **9.7.3 Program/Erase Voltage Signal Conditioning** for information on maintaining the proper relationship between V_{FPE} and V_{DD} during flash erasure.

1. Raise the voltage applied to the to the V_{FPE} pin from V_{DD} (the normal read level) to the program/erase level. Accurate V_{FPE} control is best accomplished with a dedicated flash program/erase voltage generator integrated circuit.
2. Set the LAT and ERAS bits in BFECTL; leave all other BFECTL bits clear. This configures the module for erasure.
3. Using the information provided in **Table 9-2**, write to an array address that will erase the desired BEFLASH block or to an array address or control register that will bulk erase the BEFLASH array. The data written does not matter as this write is part of the interlock sequence required for erasure.
4. Set the LAT, ERAS, and ENPE bits in BFECTL. This starts the erase pulse.
5. Allow the erase pulse time to elapse. The erase pulse time is specified by the t_{epk} parameter in **APPENDIX A ELECTRICAL CHARACTERISTICS.**
6. Keeping LAT and ERAS set, clear ENPE in BFECTL. This stops the erase pulse.
7. Allow the erase recovery time delay to elapse while the high voltage to the array is shut off. The program recovery time is specified by the parameter t_{er} in **APPENDIX A ELECTRICAL CHARACTERISTICS**.
8. If bulk erasing the BEFLASH, read the entire array and its control registers to verify erasure. For block erase operations, it is only necessary to verify erasure of the specified block, as all other BEFLASH blocks and the block-erasable flash control register shadow bits will not change. The erased state of a flash array word is $FFFF. The erased states of control registers with shadow bits are listed in **APPENDIX D REGISTER SUMMARY**. When reading a flash control register with shadow bits, the LAT bit in BFECTL must be set or the read will return the contents of the control register and not the state of its shadow bits.
9. If appropriate array locations (and control registers with shadow bits if bulk erasing) are not erased, calculate the new erase pulse time as $t_{epk} = t_{ei}$ (erase pulse increment) $\times$ erase pulse number. Repeat steps 3 through 8 until all locations erase or until the maximum number of erases pulses has been applied. The maximum number of erase pulses is specified by the parameter n_{ep} in **APPENDIX A ELECTRICAL CHARACTERISTICS**.
10. When all appropriate locations are erased, calculate the erase margin pulse time, t_{epm} (the summation of all previous erase pulses) and repeat steps 3 through 8 for the single margin pulse. This applies 100% erase margin to the array.
11. Clear the LAT and ERAS bits in BFECTL. This permits normal access to the array.
12. Lower the voltage applied to the V_{FPE} pin from the program/erase level to V_{DD} (the normal read level).



NOTES:
1. SEE APPENDIX A ELECTRICAL CHARACTERISTICS FOR V_{FPE} PIN VOLTAGE SEQUENCING.
2. THE MARGIN FLAG IS A SOFTWARE-DEFINED FLAG THAT INDICATES WHETHER THE PROGRAM SEQUENCE IS GENERATING ERASE PULSES OR MARGIN PULSES.

Figure 9-2 Erasure Flowchart

9.7.3 Program/Erase Voltage Signal Conditioning

A voltage of at least V_{DD} - 10% must be applied to the V_{FPE} pins at all times, or damage to the flash EEPROM module can occur as a result of power-up and power-down transients. V_{FPE} also must not rise to the program/erase level before V_{DD} reaches its minimum operating level. **Figure 9-3** shows the required operating envelope for V_{DD} and V_{FPE}. Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for V_{FPE} maximum and operating specifications.

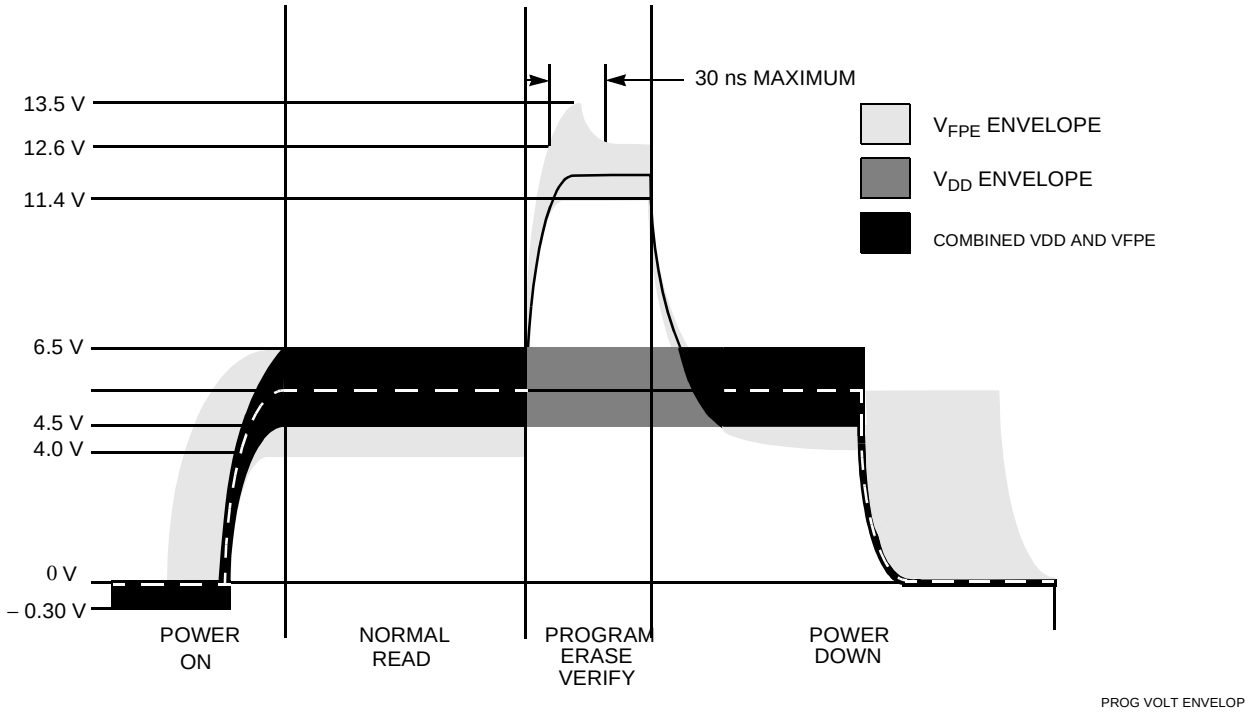


Figure 9-3 Programming Voltage Envelope

Use of a dedicated circuit to generate the 12.0V ± 5% required for program and erase operations is recommended. In addition, the V_{FPE} conditioning/protection circuit shown in **Figure 9-4** should be used to maintain proper voltage levels on the V_{FPE} pin and to protect it from power-up and power-down transients.

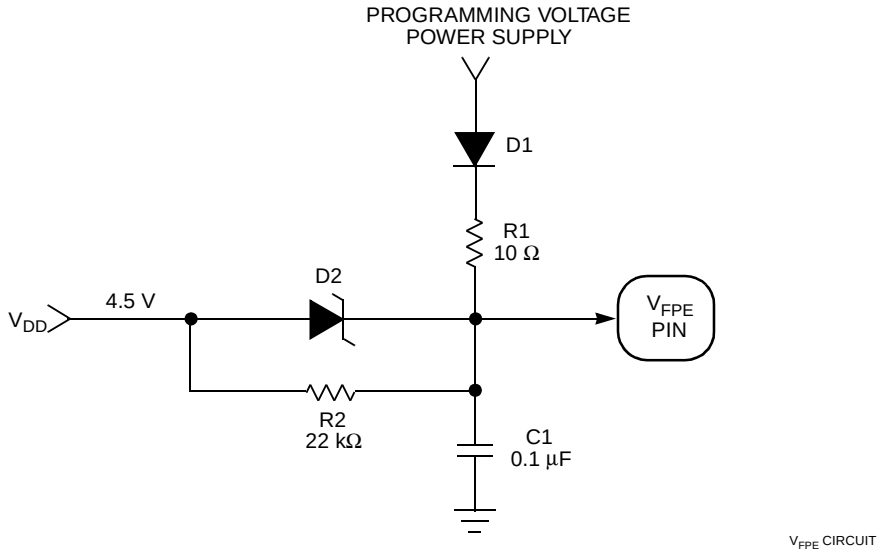


Figure 9-4 V_{FPE} Conditioning Circuit

V_{FPE} is pulled up to V_{DD} via Schottky diode D2. Application of the programming voltage via diode D1 reverse-biases D2, protecting V_{DD} from excessive reverse current. D2 also protects the flash module from damage should the programming voltage fall to zero. The programming voltage power supply should be adjusted to compensate for the forward-bias drop across D1. The charge time constant of R1 and C1 filters transients, while R2 provides a discharge bleed path for C1. Allow for RC charge and discharge times when applying and removing system power. When using this circuit, keep leakage from external devices connected to the V_{FPE} pin low to minimize diode voltage drop.

SECTION 10
ANALOG-TO-DIGITAL CONVERTER

This section is an overview of the analog-to-digital converter module (ADC). Refer to the *ADC Reference Manual* (ADCRM/AD) for a comprehensive discussion of ADC capabilities. Refer to APPENDIX A ELECTRICAL CHARACTERISTICS for ADC timing and electrical specifications. Refer to **APPENDIX D REGISTER SUMMARY** for register address mapping and bit/field definitions.

10.1 General

The ADC is a unipolar, successive-approximation converter with eight modes of operation. It has selectable 8 or 10-bit resolution. Monotonicity is guaranteed in both modes.

A bus interface unit handles communication between the ADC and other microcontroller modules, and supplies IMB timing signals to the ADC. Special operating modes and test functions are controlled by a module configuration register (ADCMCR) and a factory test register (ADCTST).

ADC module conversion functions can be grouped into three basic subsystems: an analog front end, a digital control section, and result storage. **Figure 10-1** is a functional block diagram of the ADC module.

In addition to use as multiplexer inputs, the eight analog inputs can be used as a general-purpose digital input port (port ADA), provided signals are within logic level specification. A port data register (PORTADA) is used to access input data.

10.2 External Connections

The ADC uses 12 pins on the MCU package. Eight pins are analog inputs (which can also be used as digital inputs), two pins are dedicated analog reference connections (V_{RH} and V_{RL}), and two pins are analog supply connections (V_{DDA} and V_{SSA}).

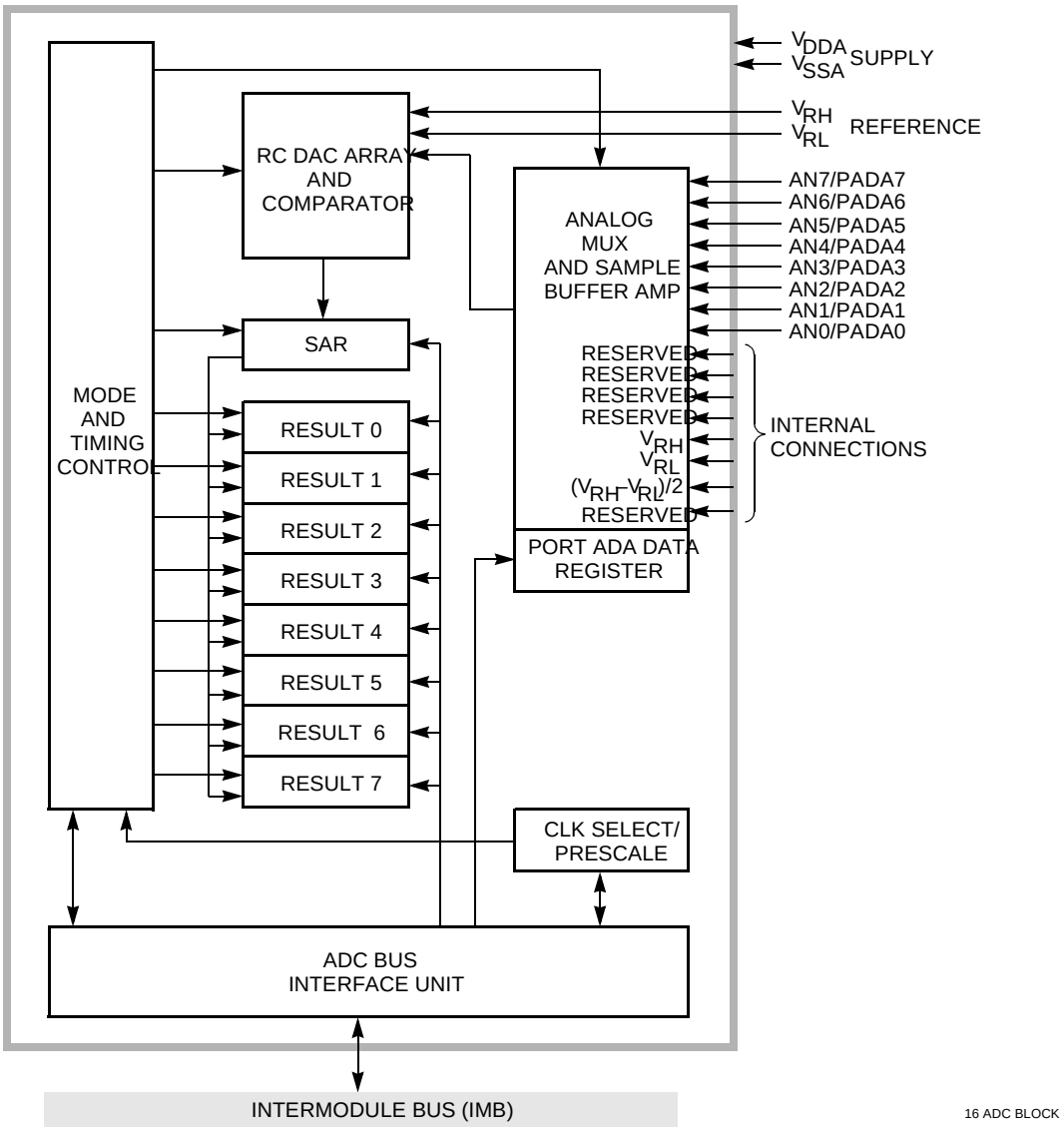


Figure 10-1 ADC Block Diagram

10.2.1 Analog Input Pins

Each of the eight analog input pins(AN[7:0]) is connected to a multiplexer in the ADC. The multiplexer selects an analog input for conversion to digital data.

Analog input pins can also be read as digital inputs, provided the applied voltage meet V_{IH} and V_{IL} specification. When used as digital inputs, the pins are organized into an 8-bit port (PORTADA), and referred to as PADA[7:0]. There is no data direction register because these pins are input only.

10.2.2 Analog Reference Pins

Separate high (V_{RH}) and low (V_{RL}) analog reference voltages are connected to the analog reference pins. The pins permit connection of regulated and filtered supplies that allow the ADC to achieve its highest degree of accuracy.

10.2.3 Analog Supply Pins

Pins V_{DDA} and V_{SSA} supply power to analog circuitry associated with the RC DAC. Other circuitry in the ADC is powered from the digital power bus (pins V_{DD} and V_{SS}). Dedicated analog power supplies are necessary to isolate sensitive ADC circuitry from noise on the digital power bus.

10.3 Programmer's Model

The ADC module is mapped into 32 words of address space. Five words are control/status registers, one word is digital port data, and 24 words provide access to the conversion results (eight addresses for each type of converted data). Two words are reserved for expansion.

Internally, the ADC has both a differential data bus and a buffered IMB data bus. Registers not directly associated with conversion functions, such as the module configuration register, the module test register, and the port data register, reside on the buffered bus, while conversion registers and result registers reside on the differential bus.

Registers that reside on the buffered bus are updated immediately when written. However, writes to ADC control registers abort any conversion in progress.

10.4 Low-Power Stop Mode

When the STOP bit in ADCMCR is set, the IMB clock signal to the ADC is disabled. This places the module in an idle state, and power consumption is minimized. The ABIU does not shut down and ADC registers are still accessible. If a conversion is in progress when STOP is set, it is aborted.

STOP is set during system reset, and must be cleared before the ADC can be used. Because analog circuit bias currents are turned off during low-power stop mode, the ADC requires recovery time after STOP is cleared.

Execution of the CPU16 LPSTOP command places the entire MCU in low-power stop mode. Refer to **5.3.9 Low-Power Operation** for more information.

10.5 Freeze Assertion Response

When the CPU16 enters background debug mode, the FREEZE signal is asserted. ADC response to FREEZE assertion is determined by the value of FRZ[1:0] in ADCMCR. **Table 10-1** shows the encoding of the FRZ field.

Table 10-1 FRZ Field Encoding

FRZ[1:0]	Response
00	Ignore FREEZE
01	Reserved
10	Finish conversion, then freeze
11	Freeze immediately

When the ADC freezes, the ADC clock stops and all sequential activity ceases. Contents of control and status registers remain valid while frozen. When the FREEZE signal is negated, the ADC activity resumes.

If the ADC freezes during a conversion, activity resumes with the next step in the conversion sequence. However, capacitors in the analog conversion circuitry will discharge while the ADC is frozen, and as a result, the conversion will be inaccurate.

Refer to **4.14.4 Background Debug Mode** for more information.

10.6 Analog Subsystem

The analog subsystem consists of a multiplexer, sample capacitors, a buffer amplifier, an RC DAC array, and a high-gain comparator. Comparator output sequences the successive approximation register (SAR). The interface between the comparator and the SAR is the boundary between ADC analog and digital subsystems.

10.6.1 Multiplexer

The multiplexer selects one of 16 sources for conversion. Eight sources are internal and eight are external. Multiplexer operation is controlled by the channel selection field CD:CA in the ADCTL1 register. **Table 10-2** shows the different multiplexer channel sources. The multiplexer contains positive and negative stress protection circuitry. This circuitry prevents voltages on other input channels from affecting the current conversion.

Table 10-2 Multiplexer Channel Sources

[CD:CA] Value	Input Source
0000	AN0
0001	AN1
0010	AN2
0011	AN3
0100	AN4
0101	AN5
0110	AN6
0111	AN7
1000	Reserved
1001	Reserved
1010	Reserved
1011	Reserved
1100	V_{RH}
1101	V_{RL}
1110	$(V_{RH} - V_{RL}) / 2$
1111	Test/Reserved

10.6.2 Sample Capacitor and Buffer Amplifier

Each of the eight external input channels is associated with a sample capacitor and share a single sample buffer amplifier. After a conversion is initiated, the multiplexer output is connected to the sample capacitor at the input of the sample buffer amplifier for the first two ADC clock cycles of the sampling period. The sample amplifier buffers the input channel from the relatively large capacitance of the RC DAC array.

During the second two clock cycles of the sampling period, the sample capacitor is disconnected from the multiplexer, and the sample buffer amplifier charges the RC DAC array with the value stored in the sample capacitor.

During the third portion of the sampling period, both sample capacitor and buffer amplifier are bypassed, and the multiplexer input charges the DAC array directly. The length of this third portion of the sampling period is determined by the value of the STS field in ADCTL0.

10.6.3 RC DAC Array

The RC DAC array consists of binary-weighted capacitors and a resistor-divider chain. The array performs two functions: it acts as a sample hold circuit during conversion, and it provides each successive digital-to-analog comparison voltage to the comparator. Conversion begins with MSB comparison and ends with LSB comparison. Array switching is controlled by the digital subsystem.

10.6.4 Comparator

The comparator indicates whether each approximation output from the RC DAC array during resolution is higher or lower than the sampled input voltage. Comparator output is fed to the digital control logic, which sets or clears each bit in the successive approximation register in sequence, MSB first.

10.7 Digital Control Subsystem

The digital control subsystem includes control and status registers, clock and prescaler control logic, channel and reference select logic, conversion sequence control logic, and the successive approximation register.

The subsystem controls the multiplexer and the output of the RC array during sample and conversion periods, stores the results of comparison in the successive-approximation register, then transfers results to the result registers.

10.7.1 Control/Status Registers

There are two control registers (ADCTL0, ADCTL1) and one status register (ADSTAT). ADCTL0 controls conversion resolution, sample time, and the clock prescaler value. ADCTL1 controls analog input selection, conversion mode, and conversion initiation. A write to ADCTL0 aborts the current conversion sequence and halts the ADC. Conversion must be restarted by writing to ADCTL1. A write to ADCTL1 aborts the current conversion sequence and starts a new sequence with parameters altered by the write. ADSTAT shows conversion sequence status, conversion channel status, and conversion completion status.

The following paragraphs are a general discussion of control function. **APPENDIX D REGISTER SUMMARY** shows the ADC address map and discusses register bits and fields.

10.7.2 Clock and Prescaler Control

The ADC clock is derived from the system clock by a programmable prescaler. The ADC clock period is determined by the value of the PRS field in ADCTL0.

The prescaler has two stages. The first stage is a 5-bit modulus counter. It divides the system clock by any value from 2 to 32. The second stage is a divide-by-two circuit. ADC clock frequency is governed by the following equation:

$$f_{ADCCLK} = \frac{f_{sys}}{2 \times (PRS[4:0] + 1)}, \quad \%00001 \leq PRS[4:0] \leq \%11111$$

The ADC clock speed must be between 0.5 MHz and 2.1 MHz. The reset value of the PRS field is %00011, which divides a nominal 16.78 MHz system clock by eight, yielding the maximum ADC clock frequency. There are a minimum of four IMB clock cycles for each ADC clock cycle. The PRS[4:0] = %00000 setting is reserved and must not be used.

10.7.3 Sample Time

The first two portions of all sample periods require four ADC clock cycles. During the third portion of a sample period, the selected channel is connected directly to the RC DAC array for a specified number of clock cycles. The value of the STS field in ADCTL0 determines the number of cycles. Refer to **Table 10-3**. The number of clock cycles required for a sample period is the value specified by STS plus four. Sample time is determined by the value of PRS[4:0].

Table 10-3 Sample Time Selection

STS[1:0]	Sample Time
00	2 ADC Clock Periods
01	4 ADC Clock Periods
10	8 ADC Clock Periods
11	16 ADC Clock Periods

10.7.4 Resolution

The ADC resolution can be either eight or ten bits. Resolution is determined by the state of the RES10 bit in ADCTL0. Both 8-bit and 10-bit conversion results are automatically aligned in the result registers.

10.7.5 Conversion Control Logic

Analog-to-digital conversions are performed in sequences. Sequences are initiated by any write to ADCTL1. A write to either ADCTL0 or ADCTL1 will abort any conversion sequence already in progress and will reset the SCF and CCF flags in ADSTAT. ADCTL1 allows the selection of eight conversion modes. Each conversion mode affects the bits in ADSTAT differently. Result storage differs from mode to mode.

10.7.5.1 Conversion Parameters

Table 10-4 describes the conversion parameters controlled by bits in ADCTL1.

Table 10-4 Conversion Parameters Controlled by ADCTL1

Conversion Parameter	Description
Conversion channel	The value of the channel selection field (CD:CA) in ADCTL1 determines which multiplexer inputs are used in a conversion sequence. There are 16 possible inputs. Eight inputs are external pins (AN[7:0]), and eight are internal.
Length of sequence	A conversion sequence consists of either four or eight conversions. The number of conversions in a sequence is determined by the state of the S8CM bit in ADCTL1.
Single or continuous conversion	Conversion can be limited to a single sequence or a sequence can be performed continuously. The state of the SCAN bit in ADCTL1 determines whether a single or continuous conversion sequence is performed.
Single or multiple channel conversion	Conversion sequence(s) can be run on a single channel or on a block of four or eight channels. Conversion of multiple or single channels is controlled by the state of the MULT bit in ADCTL1.

10.7.5.2 Conversion Modes

Conversion modes are defined by the state of the SCAN, MULT, and S8CM bits in ADCTL1. **Table 10-5** shows mode numbering.

Table 10-5 ADC Conversion Modes

SCAN	MULT	S8CM	Mode
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

The following paragraphs describe each type of conversion mode:

Mode 0 — A single four-conversion sequence is performed on a single input channel specified by the value in CD:CA. Each result is stored in a separate result register (RSLT0 to RSLT3). The appropriate CCF bit in ADSTAT is set as each register is filled. The SCF bit in ADSTAT is set when the conversion sequence is complete.

Mode 1 — A single eight-conversion sequence is performed on a single input channel specified by the value in CD:CA. Each result is stored in a separate result register (RSLT0 to RSLT7). The appropriate CCF bit in ADSTAT is set as each register is filled. The SCF bit in ADSTAT is set when the conversion sequence is complete.

Mode 2 — A single conversion is performed on each of four sequential input channels, starting with the channel specified by the value in CD:CA. Each result is stored in a separate result register (RSLT0 to RSLT3). The appropriate CCF bit in ADSTAT is set as each register is filled. The SCF bit in ADSTAT is set when the last conversion is complete.

Mode 3 — A single conversion is performed on each of eight sequential input channels, starting with the channel specified by the value in CD:CA. Each result is stored in a separate result register (RSLT0 to RSLT7). The appropriate CCF bit in ADSTAT is set as each register is filled. The SCF bit in ADSTAT is set when the last conversion is complete.

Mode 4 — Continuous four-conversion sequences are performed on a single input channel specified by the value in CD:CA. Each result is stored in a separate result register (RSLT0 to RSLT3). Previous results are overwritten when a sequence repeats. The appropriate CCF bit in ADSTAT is set as each register is filled. The SCF bit in ADSTAT is set when the first four-conversion sequence is complete.

Mode 5 — Continuous eight-conversion sequences are performed on a single input channel specified by the value in CD:CA. Each result is stored in a separate result register (RSLT0 to RSLT7). Previous results are overwritten when a sequence repeats. The appropriate CCF bit in ADSTAT is set as each register is filled. The SCF bit in ADSTAT is set when the first eight-conversion sequence is complete.

Mode 6 — Continuous conversions are performed on each of four sequential input channels, starting with the channel specified by the value in CD:CA. Each result is stored in a separate result register (RSLT0 to RSLT3). The appropriate CCF bit in ADSTAT is set as each register is filled. The SCF bit in ADSTAT is set when the first four-conversion sequence is complete.

Mode 7 — Continuous conversions are performed on each of eight sequential input channels, starting with the channel specified by the value in CD:CA. Each result is stored in a separate result register (RSLT0 to RSLT7). The appropriate CCF bit in ADSTAT is set as each register is filled. The SCF bit in ADSTAT is set when the first eight-conversion sequence is complete.

Table 10-6 summarizes ADC operation when MULT is cleared (single channel modes). **Table 10-7** summarizes ADC operation when MULT is set (multi-channel modes). Number of conversions per channel is determined by SCAN. Channel numbers are given in order of conversion.

Table 10-6 Single-Channel Conversions (MULT = 0)

S8CM	CD	CC	CB	CA	Input	Result Register[1]
0	0	0	0	0	AN0	RSLT[0:3]
0	0	0	0	1	AN1	RSLT[0:3]
0	0	0	1	0	AN2	RSLT[0:3]
0	0	0	1	1	AN3	RSLT[0:3]
0	0	1	0	0	AN4	RSLT[0:3]
0	0	1	0	1	AN5	RSLT[0:3]
0	0	1	1	0	AN6	RSLT[0:3]
0	0	1	1	1	AN7	RSLT[0:3]
0	1	0	0	0	Reserved	RSLT[0:3]
0	1	0	0	1	Reserved	RSLT[0:3]
0	1	0	1	0	Reserved	RSLT[0:3]
0	1	0	1	1	Reserved	RSLT[0:3]
0	1	1	0	0	V_{RH}	RSLT[0:3]
0	1	1	0	1	V_{RL}	RSLT[0:3]
0	1	1	1	0	$(V_{RH} - V_{RL}) / 2$	RSLT[0:3]
0	1	1	1	1	Test/Reserved	RSLT[0:3]
1	0	0	0	0	AN0	RSLT[0:7]
1	0	0	0	1	AN1	RSLT[0:7]
1	0	0	1	0	AN2	RSLT[0:7]
1	0	0	1	1	AN3	RSLT[0:7]
1	0	1	0	0	AN4	RSLT[0:7]
1	0	1	0	1	AN5	RSLT[0:7]
1	0	1	1	0	AN6	RSLT[0:7]
1	0	1	1	1	AN7	RSLT[0:7]
1	1	0	0	0	Reserved	RSLT[0:7]
1	1	0	0	1	Reserved	RSLT[0:7]
1	1	0	1	0	Reserved	RSLT[0:7]
1	1	0	1	1	Reserved	RSLT[0:7]
1	1	1	0	0	V_{RH}	RSLT[0:7]
1	1	1	0	1	V_{RL}	RSLT[0:7]
1	1	1	1	0	$(V_{RH} - V_{RL}) / 2$	RSLT[0:7]
1	1	1	1	1	Test/Reserved	RSLT[0:7]

NOTES:

1. Result register (RSLT) is either RJURR, LJSRR, or LJURR, depending on the address read.

Table 10-7 Multiple-Channel Conversions (MULT = 1)

S8CM	CD	CC	CB	CA	Input	Result Register[1]
0	0	0	X	X	AN0	RSLT0
					AN1	RSLT1
					AN2	RSLT2
					AN3	RSLT3
0	0	1	X	X	AN4	RSLT0
					AN5	RSLT1
					AN6	RSLT2
					AN7	RSLT3
0	1	0	X	X	Reserved	RSLT0
					Reserved	RSLT1
					Reserved	RSLT2
					Reserved	RSLT3
0	1	1	X	X	V_{RH}	RSLT0
					V_{RL}	RSLT1
					$(V_{RH} - V_{RL}) / 2$	RSLT2
					Test/Reserved	RSLT3
1	0	X	X	X	AN0	RSLT0
					AN1	RSLT1
					AN2	RSLT2
					AN3	RSLT3
					AN4	RSLT4
					AN5	RSLT5
					AN6	RSLT6
					AN7	RSLT7
1	1	X	X	X	Reserved	RSLT0
					Reserved	RSLT1
					Reserved	RSLT2
					Reserved	RSLT3
					V_{RH}	RSLT4
					V_{RL}	RSLT5
					$(V_{RH} - V_{RL}) / 2$	RSLT6
					Test/Reserved	RSLT7

NOTES:
1. Result register (RSLT) is either RJURR, LJSRR, or LJURR, depending on the address read.

10.7.6 Conversion Timing

Total conversion time is made up of initial sample time, transfer time, final sample time, and resolution time. Initial sample time is the time during which a selected input channel is connected to the sample buffer amplifier through a sample capacitor. During transfer time, the sample capacitor is disconnected from the multiplexer, and the RC DAC array is driven by the sample buffer amplifier. During final sampling time, the sample capacitor and amplifier are bypassed, and the multiplexer input charges the RC DAC array directly. During resolution time, the voltage in the RC DAC array is converted to a digital value, and the value is stored in the SAR.

Initial sample time and transfer time are fixed at two ADC clock cycles each. Final sample time can be 2, 4, 8, or 16 ADC clock cycles, depending on the value of the STS field in ADCTL0. Resolution time is ten cycles for 8-bit conversions and twelve cycles for 10-bit conversions.

Transfer and resolution require a minimum of 16 ADC clocks (8 µs with a 2.1 MHz ADC clock) for 8-bit conversions and 18 ADC clocks (9 µs with a 2.1 MHz ADC clock) for 10-bit conversions. If the maximum final sample time (16 ADC clocks) is used, total conversion time is 15 µs for 8-bit conversions and 16 µs for 10-bit conversions (with a 2.1 MHz ADC clock).

Figures 10-2 and **10-3** illustrate the timing for 8- and 10-bit conversions, respectively. These diagrams assume a final sampling selection of two ADC clocks.

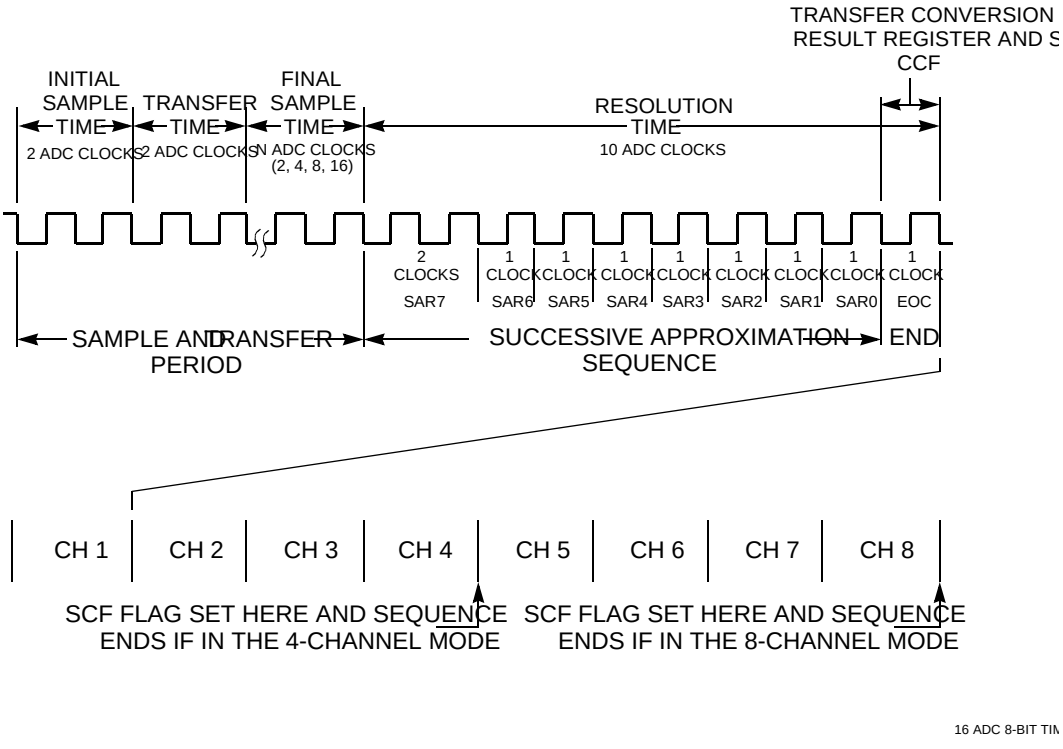


Figure 10-2 8-Bit Conversion Timing

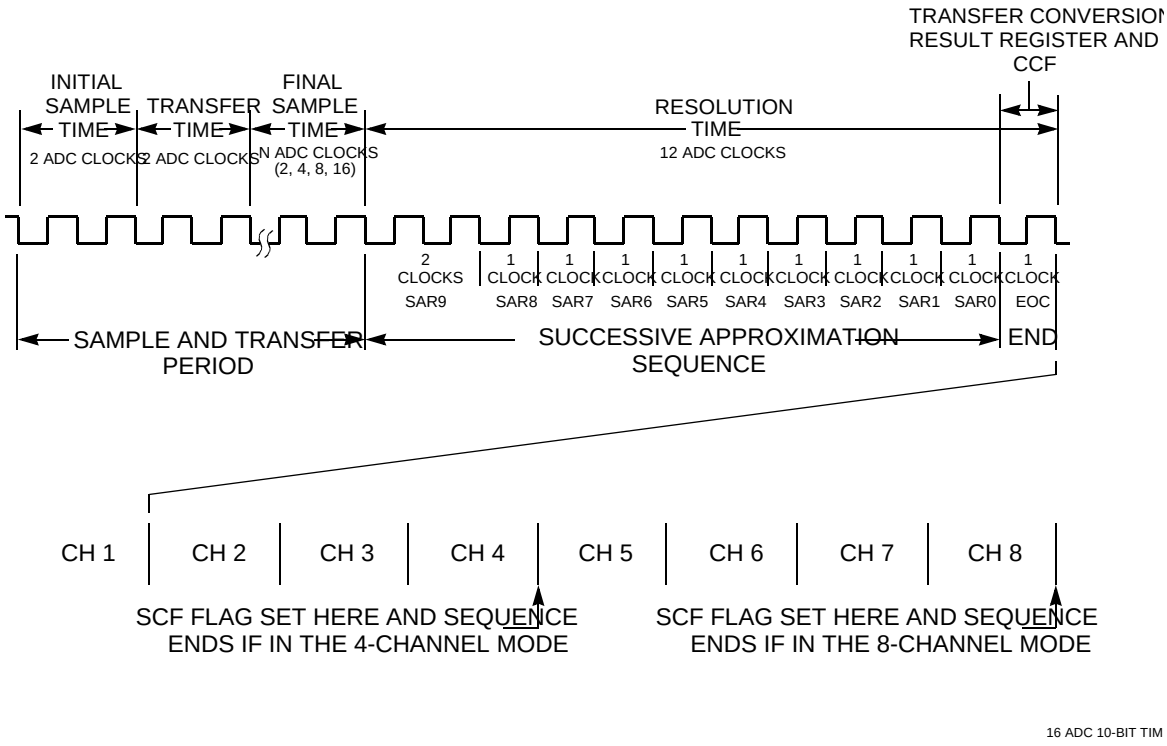


Figure 10-3 10-Bit Conversion Timing

10.7.7 Successive Approximation Register

The successive approximation register (SAR) accumulates the result of each conversion one bit at a time, starting with the most significant bit.

At the start of the resolution period, the MSB of the SAR is set, and all less significant bits are cleared. Depending on the result of the first comparison, the MSB is either left set or cleared. Each successive bit is set or left cleared in descending order until all eight or ten bits have been resolved.

When a conversion is complete, the contents of the SAR are transferred to the appropriate result register. Refer to **APPENDIX D REGISTER SUMMARY** for register mapping and configuration.

10.7.8 Result Registers

Result registers are used to store the data from completed conversions. Each register can be read from three different addresses in the ADC memory map. The format of the result data depends on the address from which it is read. **Table 10-8** shows the three types of formats.

Table 10-8 Result Register Formats

Result Data Format	Description
Unsigned right-justified	Conversion result is unsigned right-justified data. Bits [9:0] are used for 10-bit conversions and bits [7:0] are used for 8-bit conversions. Bits [15:10] for 10-bit results and bits [15:8] for 8-bit results always return zero when read.
Signed left-justified	Conversion result is signed left-justified data. Bits [15:6] are used for 10-bit conversions and bits [15:8] are used for 8-bit conversions. Although the ADC is unipolar, it is assumed that the zero point is ($V_{RH} - V_{RL}$) / 2 when this format is used. The value read from the register is an offset two's-complement number; for positive input, bit 15 equals zero, for negative input, bit 15 equals one. Bits [5:0] for 10-bit results and bits [7:0] for 8-bit results always return zero when read.
Unsigned left-justified	Conversion result is unsigned left-justified data. Bits [15:6] are used for 10-bit conversions and bits [15:8] are used for 8-bit conversions. Bits [5:0] for 10-bit results and bits [7:0] for 8-bit results always return zero when read.

10.8 Pin Considerations

The ADC requires accurate, noise-free input signals for proper operation. The following sections discuss the design of external circuitry to maximize ADC performance.

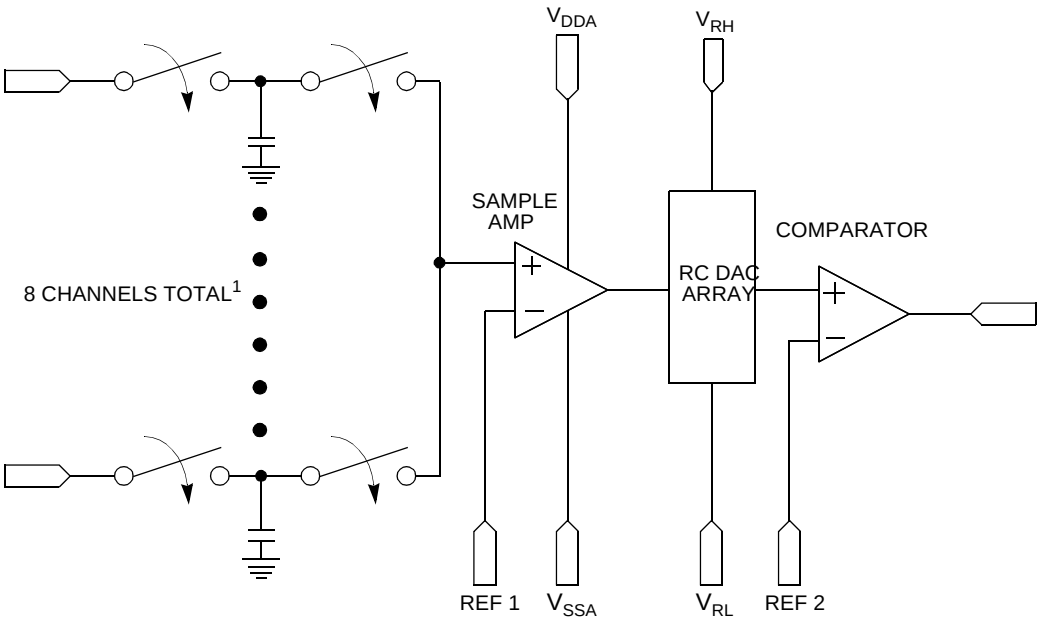
10.8.1 Analog Reference Pins

No A/D converter can be more accurate than its analog reference. Any noise in the reference can result in at least that much error in a conversion. The reference for the ADC, supplied by pins V_{RH} and V_{RL}, should be low-pass filtered from its source to obtain a noise-free, clean signal. In many cases, simple capacitive bypassing may suffice. In extreme cases, inductors or ferrite beads may be necessary if noise or RF energy is present. Series resistances connected to V_{RH} and V_{RL} are not advisable since the internal resistor string in the RC DAC array effectively requires a DC current requirement from the reference voltage. Any external resistance may introduce error under certain conditions. Any series devices in the filter network should contain a minimum amount of DC resistance.

For accurate conversion results, the analog reference voltages must be within the limits defined by V_{DDA} and V_{SSA}, as explained in the following subsection.

10.8.2 Analog Power Pins

The analog supply pins (V_{DDA} and V_{SSA}) define the limits of the analog reference voltages (V_{RH} and V_{RL}) and of the analog multiplexer inputs. **Figure 10-4** is a diagram of the analog input circuitry.



NOTES:
1. TWO SAMPLE AMPS EXIST ON THE ADC WITH 8 CHANNELS ON EACH SAMPLE AMP.

ADC 8CH SAMPLE AMI

Figure 10-4 Analog Input Circuitry

Since the sample amplifier is powered by V_{DDA} and V_{SSA}, it can accurately transfer input signal levels up to but not above V_{DDA} and down to but not below V_{SSA}. If the input signal is outside of this range, the output from the sample amplifier is clipped.

In addition, V_{RH} and V_{RL} must be within the range defined by V_{DDA} and V_{SSA}. As long as $V_{RH} \leq V_{DDA}$ and $V_{RL} \geq V_{SSA}$ and the sample amplifier has accurately transferred the input signal, resolution is ratiometric within the limits defined by V_{RL} and V_{RH}. If $V_{RH} \geq V_{DDA}$, the sample amplifier can never transfer a full-scale value. If $V_{RL} \leq V_{SSA}$, the sample amplifier can never transfer a zero value.

Figure 10-5 shows the results of reference voltages outside the range defined by V_{DDA} and V_{SSA}. At the top of the input signal range, V_{DDA} is 10 mV lower than V_{RH}. This results in a maximum obtainable 10-bit conversion value of $03FE. At the bottom of the signal range, V_{SSA} is 15 mV higher than V_{RL}, resulting in a minimum obtainable 10-bit conversion value of $0003.

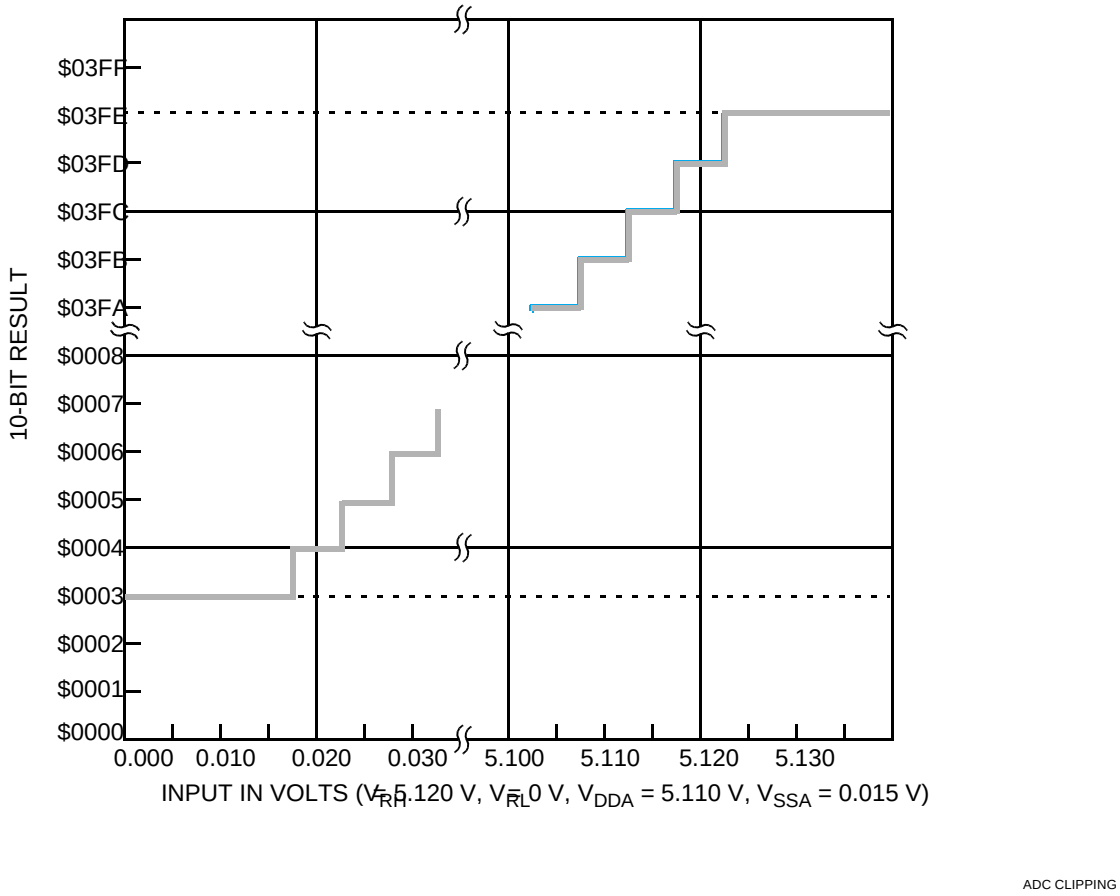


Figure 10-5 Errors Resulting from Clipping

10.8.3 Analog Supply Filtering and Grounding

Two important factors influencing performance in analog integrated circuits are supply filtering and grounding. Generally, digital circuits use bypass capacitors on every V_{DD}/V_{SS} pin pair. This applies to analog subsystems or submodules also. Equally important as bypassing, is the distribution of power and ground.

Analog supplies should be isolated from digital supplies as much as possible. This necessity stems from the higher performance requirements often associated with analog circuits. Therefore, deriving an analog supply from a local digital supply is not recommended. However, if for economic reasons digital and analog power are derived from a common regulator, filtering of the analog power is recommended in addition to the bypassing of the supplies already mentioned. For example, an RC low pass filter could be used to isolate the digital and analog supplies when generated by a common regulator. If multiple high precision analog circuits are locally employed (such as two A/D converters), the analog supplies should be isolated from each other as sharing supplies introduces the potential for interference between analog circuits.

Grounding is the most important factor influencing analog circuit performance in mixed signal systems (or in stand alone analog systems). Care must be taken to avoid the introduction of additional noise sources into the analog circuitry. Common sources of noise include ground loops, inductive coupling, and the inappropriate combination of digital and analog grounds.

The problem of how and when to combine digital and analog grounds arises from the large transients which the digital ground must handle. If the digital ground is not able to handle large transients, the current from them can return to ground through the analog ground.The excess current overflowing from the digital ground into the analog ground causes performance degradation by developing a differential voltage between the true analog ground and the digital ground pin. Skewed analog results occur because the ground observed by the analog circuit is no longer true ground.

Two similar approaches designed to improve or eliminate the problems associated with grounding excess transient currents involve star-point ground systems. One approach is to star-point the different grounds at the power supply origin, thus keeping each ground isolated. Refer to **Figure 10-6**.

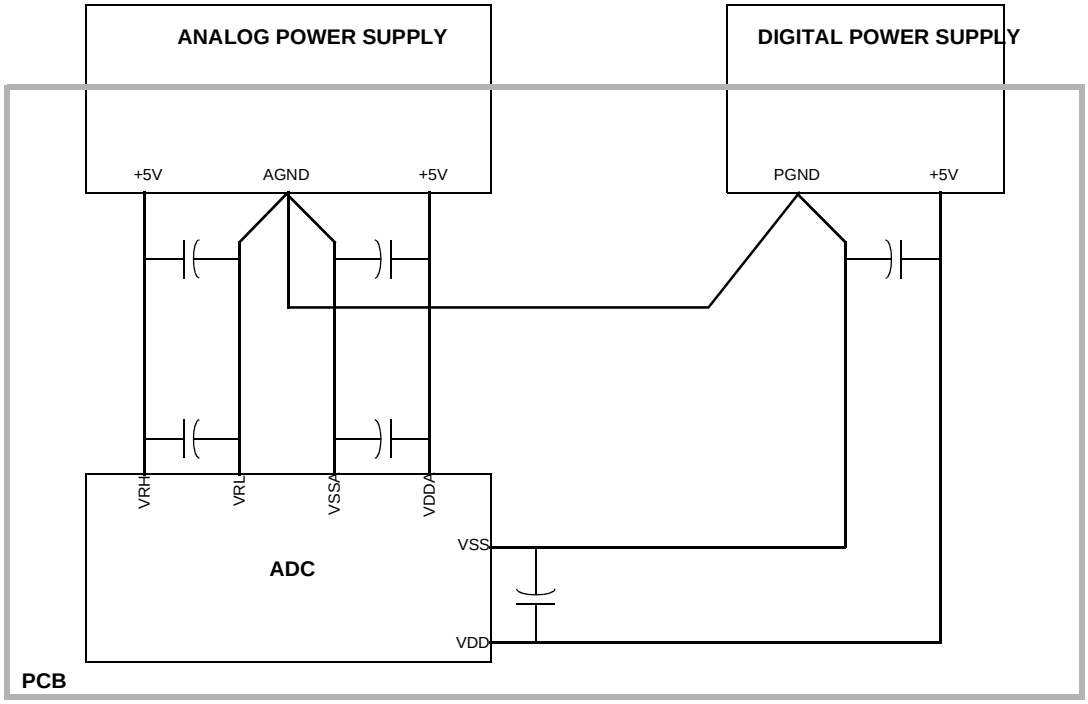


Figure 10-6 Star-Ground at the Point of Power Supply Origin

Another approach is to star-point the different grounds near the analog ground pin on the microcontroller by using small traces for connecting the non-analog grounds to the analog ground. The small traces are meant only to accommodate DC differences, not AC transients.

NOTE

This star-point scheme still requires adequate grounding for digital and analog subsystems in addition to the star-point ground.

Other suggestions for PCB layout in which the ADC is employed include the following:

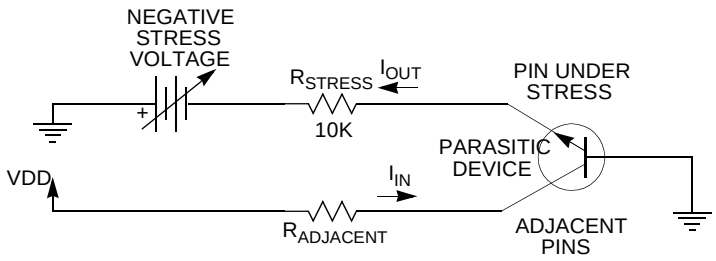
- The analog ground must be low impedance to all analog ground points in the circuit.
- Bypass capacitors should be as close to the power pins as possible.
- The analog ground should be isolated from the digital ground. This can be done by cutting a separate ground plane for the analog ground.
- Non-minimum traces should be utilized for connecting bypass capacitors and filters to their corresponding ground/power points.
- Keep distances at a minimum for trace runs when possible.

10.8.4 Accommodating Positive/Negative Stress Conditions

Positive or negative stress refers to conditions which exceed nominally defined operating limits. Examples include applying a voltage exceeding the normal limit on an input (for example, voltages outside of the suggested supply/reference ranges) or causing currents into or out of the pin which exceed normal limits. ADC specific considerations are voltages $> V_{DDA}$ and V_{RH} or $< V_{SSA}$ and V_{RL} applied to an analog input which cause excessive currents into or out of the input. Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for exact magnitudes.

Both stress conditions can potentially disrupt conversion results on neighboring inputs. Parasitic devices associated with CMOS processes can cause an immediate disruptive influence on neighboring pins. Common examples of parasitic devices are substrate (ground) diodes and bipolar devices with the base terminal tied to substrate ground. Under stress conditions, current introduced on an adjacent pin can cause errors on adjacent channels by developing a voltage drop across the adjacent external channel source impedances.

Figure 10-7 shows an active parasitic bipolar when an input pin is subjected to negative stress conditions. Positive stress conditions do not activate a similar parasitic device.



ADC PAR STRESS CONI

Figure 10-7 Input Pin Subjected to Negative Stress

The current out of the pin (I_{OUT}) under negative stress is determined by the following equation:

$$I_{OUT} = \frac{|V_{STRESS} - V_{BE}|}{R_{STRESS}}$$

where:

V_{STRESS} = Adjustable voltage source

V_{BE} = Parasitic bipolar base/emitter voltage (refer to $V_{NEGCLAMP}$ in **APPENDIX A ELECTRICAL CHARACTERISTICS**.

R_{STRESS} = Source impedance (10K resistor in **Figure 10-7** on stressed channel)

The current into (I_{IN}) the neighboring pin is determined by the $1/K_N$ (Gain) of the parasitic bipolar transistor ($1/K_N << 1$).

One way to minimize the impact of stress conditions on the ADC is to apply voltage limiting circuits such as diodes to supply and ground. However, leakage from such circuits and the potential influence on the sampled voltage to be converted must be considered. Refer to **Figure 10-8**.

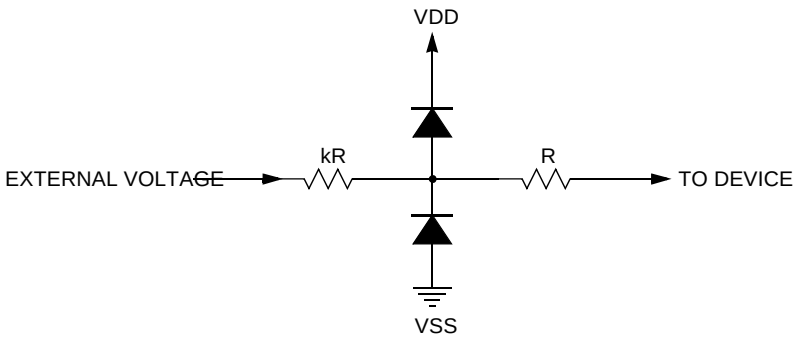


Figure 10-8 Voltage LImiting Diodes in a Negative Stress Circuit

Another method for minimizing the impact of stress conditions on the ADC is to strategically allocate ADC inputs so that the lower accuracy inputs are adjacent to the inputs most likely to see stress conditions.

Finally, suitable source impedances should be selected to meet design goals and minimize the effect of stress conditions.

10.8.5 Analog Input Considerations

The source impedance of the analog signal to be measured and any intermediate filtering should be considered whether external multiplexing is used or not. **Figure 10-9** shows the connection of eight typical analog signal sources to one ADC analog input pin through a separate multiplexer chip. Also, an example of an analog signal source connected directly to a ADC analog input channel is displayed.

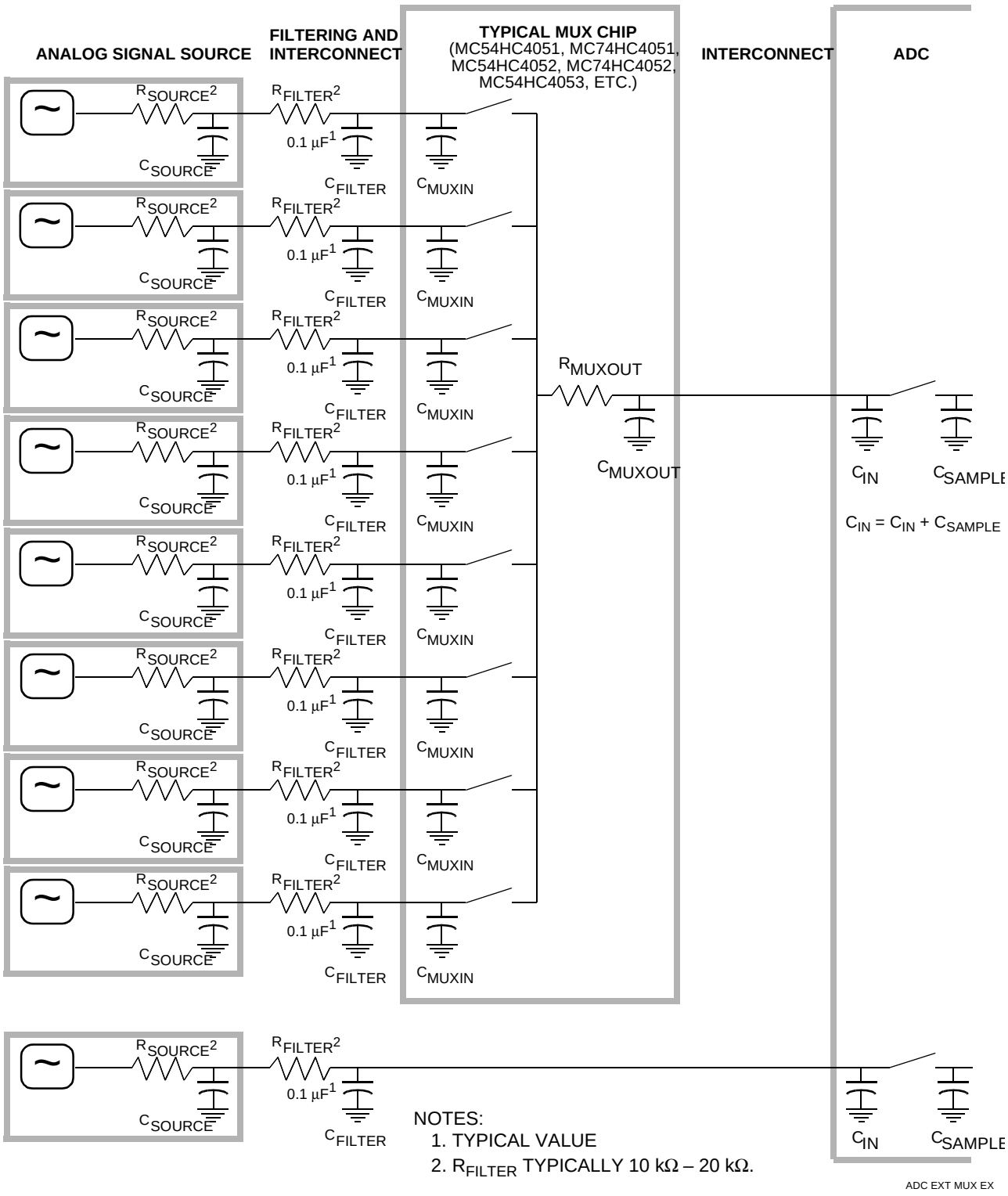


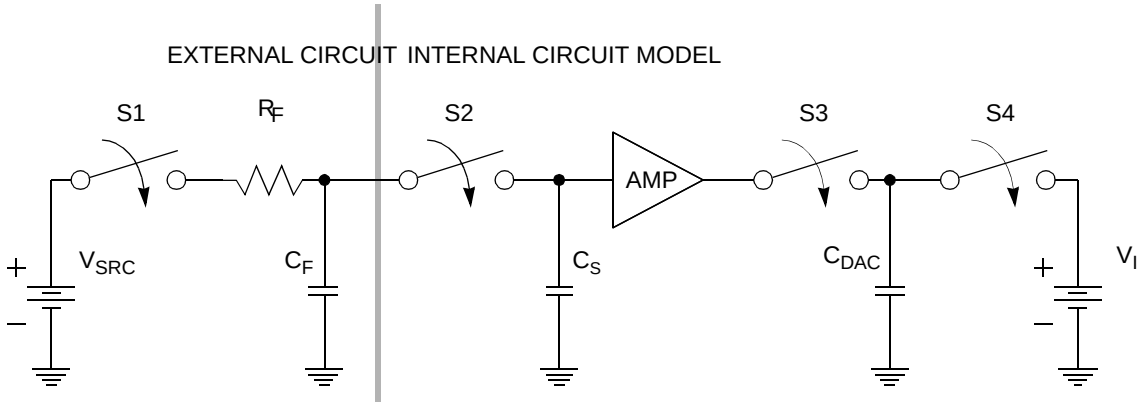
Figure 10-9 External Multiplexing Of Analog Signal Sources

10.8.6 Analog Input Pins

Analog inputs should have low AC impedance at the pins. Low AC impedance can be realized by placing a capacitor with good high frequency characteristics at the input pin of the part. Ideally, that capacitor should be as large as possible (within the practical range of capacitors that still have good high frequency characteristics). This capacitor has two effects. First, it helps attenuate any noise that may exist at the input. Second, it sources charge during the sample period when the analog signal source is a high-impedance source.

Series resistance can be used with the capacitor on an input pin to implement a simple RC filter. The maximum level of filtering at the input pins is application dependent and is based on the bandpass characteristics required to accurately track the dynamic characteristics of an input. Simple RC filtering at the pin may be limited by the source impedance of the transducer or circuit supplying the analog signal to be measured. Refer to **10.8.6.2 Error Resulting from Leakage**. In some cases, the size of the capacitor at the pin may be very small.

Figure 10-10 is a simplified model of an input channel. Refer to this model in the following discussion of the interaction between external circuitry and the circuitry inside the ADC.



V_{SRC} = SOURCE VOLTAGE
R_F = FILTER IMPEDANCE (SOURCE IMPEDANCE INCLUDED)
C_F = FILTER CAPACITOR
C_S = INTERNAL CAPACITANCE (FOR A BYPASSED CHANNEL, THIS IS THE C_{DAC} CAPACITANCE)
C_{DAC} = DAC CAPACITOR ARRAY
V_I = INTERNAL VOLTAGE SOURCE FOR PRECHARGE ($V_{DDA}/2$)

ADC SAMPLE AMP MOD

Figure 10-10 Electrical Model of an A/D Input Pin

In **Figure 10-10**, R_F and C_F comprise the user's external filter circuit. C_S is the internal sample capacitor. Each channel has its own capacitor. C_S is never precharged; it retains the value of the last sample. V_I is an internal voltage source used to precharge the DAC capacitor array (C_{DAC}) before each sample. The value of this supply is $V_{DDA}/2$, or 2.5 volts for 5-volt operation.

The following paragraphs provide a simplified description of the interaction between the ADC and external circuitry. This circuitry is assumed to be a simple RC low-pass filter passing a signal from a source to the ADC input pin. The following simplifying assumptions are made:

- The source impedance is included with the series resistor of the RC filter.
- The external capacitor is perfect (no leakage, no significant dielectric absorption characteristics, etc.)
- All parasitic capacitance associated with the input pin is included in the value of the external capacitor.
- Inductance is ignored.
- The “on” resistance of the internal switches is zero ohms and the “off” resistance is infinite.

10.8.6.1 Settling Time for the External Circuit

The values for R_F and C_F determine the length of time required to charge C_F to the source voltage level (V_{SRC}).

At time t = 0, S1 in **Figure 10-10** closes. S2 is open, disconnecting the internal circuitry from the external circuitry. Assume that the initial voltage across C_F is 0. As C_F charges, the voltage across it is determined by the following equation, where t is the total charge time:

$$V_{CF} = V_{SRC}(1 - e^{-t/R_F C_F})$$

When t = 0, the voltage across C_F = 0. As t approaches infinity, V_{CF} will equal V_{SRC}. (This assumes no internal leakage.) With 10-bit resolution, 1/2 of a count is equal to 1/2048 full-scale value. Assuming worst case (V_{SRC} = full scale), **Table 10-9** shows the required time for C_F to charge to within 1/2 of a count of the actual source voltage during 10-bit conversions. **Table 10-9** is based on the RC network in **Figure 10-10**.

NOTE

The following times are completely independent of the A/D converter architecture (assuming the ADC is not affecting the charging).

Table 10-9 External Circuit Settling Time (10-Bit Conversions)

Filter Capacitor (C_F)	Source Resistance (R_F)			
	100 Ω	1 kΩ	10 kΩ	100 kΩ
1 µF	760 µs	7.6 ms	76 ms	760 ms
.1 µF	76 µs	760 µs	7.6 ms	76 ms
.01 µF	7.6 µs	76 µs	760 µs	7.6 ms
.001 µF	760 ns	7.6 µs	76 µs	760 µs
100 pF	76 ns	760 ns	7.6 µs	76 µs

The external circuit described in **Table 10-9** is a low-pass filter. When measuring an AC component of the external signal, the characteristics of this filter must be taken into account.

10.8.6.2 Error Resulting from Leakage

A series resistor limits the current to a pin, therefore input leakage acting through a large source impedance can degrade A/D accuracy. The maximum input leakage current is specified in APPENDIX A ELECTRICAL CHARACTERISTICS. Input leakage is greatest at high operating temperatures and as a general rule decreases by one half for each 10 °C decrease in temperature.

Assuming $V_{RH} - V_{RL}$ = 5.12 V, 1 count (assuming 10-bit resolution) corresponds to 50 mV of input voltage. A typical input leakage of 50 nA acting through 100 kΩ of external series resistance results in an error of less than 1 count (5.0 mV). If the source impedance is 1 MΩ and a typical leakage of 50 nA is present, an error of 10 counts (50 mV) is introduced.

In addition to internal junction leakage, external leakage (e.g., if external clamping diodes are used) and charge sharing effects with internal capacitors also contribute to the total leakage current. **Table 10-10** illustrates the effect of different levels of total leakage on accuracy for different values of source impedance. The error is listed in terms of 10-bit counts.

CAUTION

Leakage of 10 nA is obtainable only within a limited temperature range.

Table 10-10 Error Resulting From Input Leakage (IOFF)

Source Impedance	Leakage Value (10-Bit Conversions)			
	10 nA	50 nA	100 nA	1000 nA
1 kΩ	—	—	—	0.2 counts
10 kΩ	—	0.1 counts	0.2 counts	2 counts
100 kΩ	0.2 counts	1 count	2 counts	20 counts

SECTION 11
MULTICHANNEL COMMUNICATION INTERFACE

This section is an overview of the multichannel communication interface (MCCI) module. Refer to the *MCCI Reference Manual* (MCCIRM/AD) for more information on MCCI capabilities. Refer to APPENDIX A ELECTRICAL CHARACTERISTICS for MCCI timing and electrical specifications. Refer to **APPENDIX D REGISTER SUMMARY** for register address mapping and bit/field definitions.

11.1 General

The MCCI contains three serial interfaces: a serial peripheral interface (SPI) and two serial communication interfaces (SCI). **Figure 11-1** is a block diagram of the MCCI.

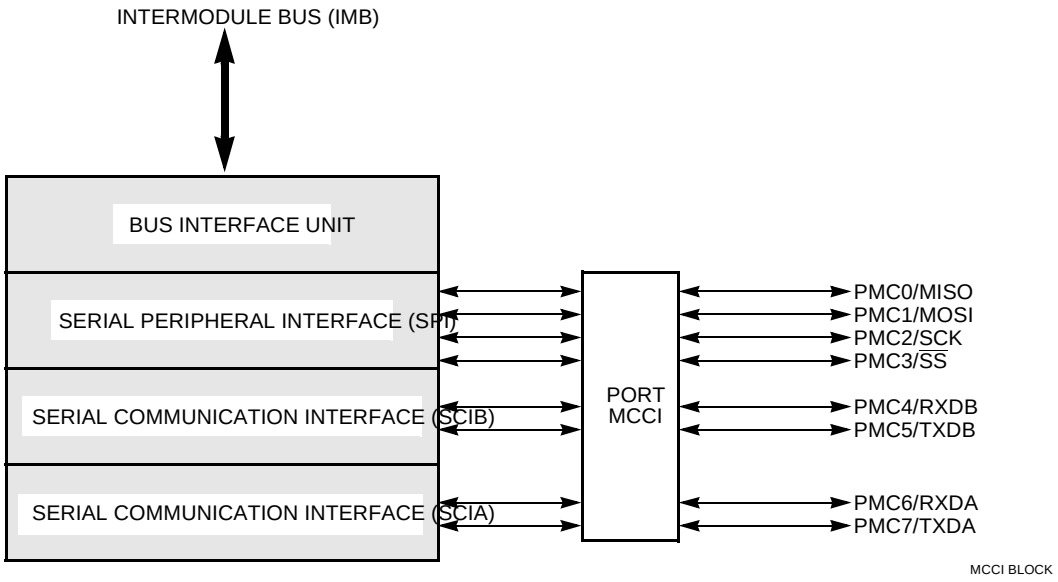


Figure 11-1 MCCI Block Diagram

The SPI provides easy peripheral expansion or interprocessor communication via a full-duplex, synchronous, three-line bus: data in, data out, and a serial clock. Serial transfer of 8 or 16 bits can begin with the most significant bit (MSB) or least significant bit (LSB). The MCCI module can be configured as a master or slave device. Clock control logic allows a selection of clock polarity and a choice of two clocking protocols to accommodate most available synchronous serial peripheral devices. When the SPI is configured as a master, software selects one of 254 different bit rates for the serial clock.

The SCI is a universal asynchronous receiver transmitter (UART) serial interface with a standard non-return to zero (NRZ) mark/space format. It operates in either full- or half-duplex mode: it contains separate transmitter- and receiver-enable bits and a double transmit buffer. A modulus-type baud rate generator provides rates from 64 baud to 524 kbaud with a 16.78-MHz system clock. Word length of either 8 or 9 bits is software selectable. Optional parity generation and detection provide either even or odd parity check capability. Advanced error detection circuitry catches glitches of up to 1/16 of a bit time in duration. Wakeup functions allow the CPU to run uninterrupted until meaningful data is received.

11.2 MCCI Registers and Address Map

The MCCI address map occupies 64 bytes from address $YFFC00 to $YFFC3F. It consists of MCCI global registers and SPI and SCI control, status, and data registers. Writes to unimplemented register bits have no effect, and reads of unimplemented bits always return zero.

The MM bit in the single-chip integration module 2 configuration register (SCIM2CR) defines the most significant bit (ADDR23) of the IMB address for each module. Because ADDR[23:20] are driven to the same bit as ADDR19, MM must be set to one. If MM is cleared, IMB modules are inaccessible. Refer to **5.2.1 Module Mapping** for more information about how the state of MM affects the system.

11.2.1 MCCI Global Registers

The MCCI module configuration register (MMCR) contains bits and fields to place the MCCI in low-power operation, establish the privilege level required to access MCCI registers, and establish the priority of the MCCI during interrupt arbitration. The MCCI test register (MTEST) is used only during factory test of the MCCI. The SCI interrupt level register (ILSCI) determines the level of interrupts requested by each SCI. Separate fields hold the interrupt-request levels for SCIA and SCIB. The MCCI interrupt vector register (MIVR) determines which three vectors in the exception vector table are to be used for MCCI interrupts. The SPI and both SCI interfaces have separate interrupt vectors adjacent to one another. The SPI interrupt level register (ILSPI) determines the priority level of interrupts requested by the SPI. The MCCI port data registers (PORTMC, PORTMCP) are used to configure port MCCI for general- purpose I/O. The MCCI pin assignment register (MPAR) determines which of the SPI pins (with the exception of SCK) are used by the SPI, and which pins are available for general-purpose I/O. The MCCI data direction register (DDRM) configures each pins as an input or output.

11.2.1.1 Low-Power Stop Mode

When the STOP bit in the MMCR is set, the IMB clock signal to most of the MCCI module is disabled. This places the module in an idle state and minimizes power consumption.

To ensure that the MCCI stops in a known state, assert the STOP bit before executing the CPU LPSTOP instruction. Before asserting the STOP bit, disable the SPI (clear the SPE bit) and disable the SCI receivers and transmitters (clear the RE and TE bits). Complete transfers in progress before disabling the SPI and SCI interfaces.

Once the STOP bit is asserted, it can be cleared by system software or by reset.

11.2.1.2 Privilege Levels

The supervisor bit (SUPV) in the MMCR has no effect since the CPU16 operates only in the supervisor mode.

11.2.1.3 MCCI Interrupts

The interrupt request level of each of the three MCCI interfaces can be programmed to a value of 0 (interrupts disabled) through 7 (highest priority). These levels are selected by the ILSCIA and ILSCIB fields in the SCI interrupt level register (ILSCI) and the ILSPI field in the SPI interrupt level register (ILSPI). In case two or more MCCI submodules request an interrupt simultaneously and are assigned the same interrupt request level, the SPI submodule is given the highest priority and SCIB is given the lowest.

When an interrupt is requested which is at a higher level than the interrupt mask in the CPU status register, the CPU initiates an interrupt acknowledge cycle. During this cycle, the MCCI compares its interrupt request level to the level recognized by the CPU. If a match occurs, arbitration with other modules begins.

Interrupting modules present their arbitration number on the IMB, and the module with the highest number wins. The arbitration number for the MCCI is programmed into the interrupt arbitration (IARB) field of the MMCR. Each module should be assigned a unique arbitration number. The reset value of the IARB field is $0, which prevents the MCCI from arbitrating during an interrupt acknowledge cycle. The IARB field should be initialized by system software to a value from $F (highest priority) through $1 (lowest priority). Otherwise, the CPU identifies any interrupts generated as spurious and takes a spurious-interrupt exception.

If the MCCI wins the arbitration, it generates an interrupt vector that uniquely identifies the interrupting serial interface. The six MSBs are read from the interrupt vector (INTV) field in the MCCI interrupt vector register (MIVR). The two LSBs are assigned by the MCCI according to the interrupting serial interface, as indicated in **Table 11-1**.

Table 11-1 MCCI Interrupt Vectors

Interface	INTV[1:0]
SCIA	00
SCIB	01
SPI	10

Select a value for INTV so that each MCCI interrupt vector corresponds to one of the user-defined vectors ($40–$FF). Refer to the *CPU16 Reference Manual* (CPU16RM/AD) for additional information on interrupt vectors.

11.2.2 Pin Control and General-Purpose I/O

The eight pins used by the SPI and SCI subsystems have alternate functions as general-purpose I/O pins. Configuring the MCCI submodule includes programming each pin for either general-purpose I/O or its serial interface function. In either function, each pin must also be programmed as input or output.

The MCCI data direction register (MDDR) assigns each MCCI pin as either input or output. The MCCI pin assignment register (MPAR) assigns the MOSI, MISO, and $\overline{SS}$ pins as either SPI pins or general-purpose I/O. (The fourth pin, SCK, is automatically assigned to the SPI whenever the SPI is enabled, for example, when the SPE bit in the SPI control register is set.) The receiver enable (RE) and transmitter enable (TE) bits in the SCI control registers (SCCR0A, SCCR0B) automatically assign the associated pin as an SCI pin when set or general-purpose I/O when cleared. **Table 11-2** summarizes how pin function and direction are assigned.

Table 11-2 Pin Assignments

Pin	Function Assigned By	Direction Assigned By
TXDA/PMC7	TE bit in SCCR0A	MMDR7
RXDA/PMC6	RE bit in SCCR0A	MMDR6
TXDB/PMC5	TE bit in SCCR0B	MMDR5
RXDB/PMC4	RE bit in SCCR0B	MMDR4
$\overline{SS}$/PMC3	$\overline{SS}$ bit in MPAR	MMDR3
SCK/PMC2	SPE bit in SPCR	MMDR2
MOSI/PMC1	MOSI bit in MPAR	MMDR1
MISO/PMC0	MISO bit in MPAR	MMDR0

11.3 Serial Peripheral Interface (SPI)

The SPI submodule communicates with external peripherals and other MCUs via a synchronous serial bus. The SPI is fully compatible with the serial peripheral interface systems found on other Motorola devices such as the M68HC11 and M68HC05 families. The SPI can perform full duplex three-wire or half duplex two-wire transfers. Serial transfer of 8 or 16 bits can begin with the MSB or LSB. The system can be configured as a master or slave device.

Figure 11-2 shows a block diagram of the SPI.

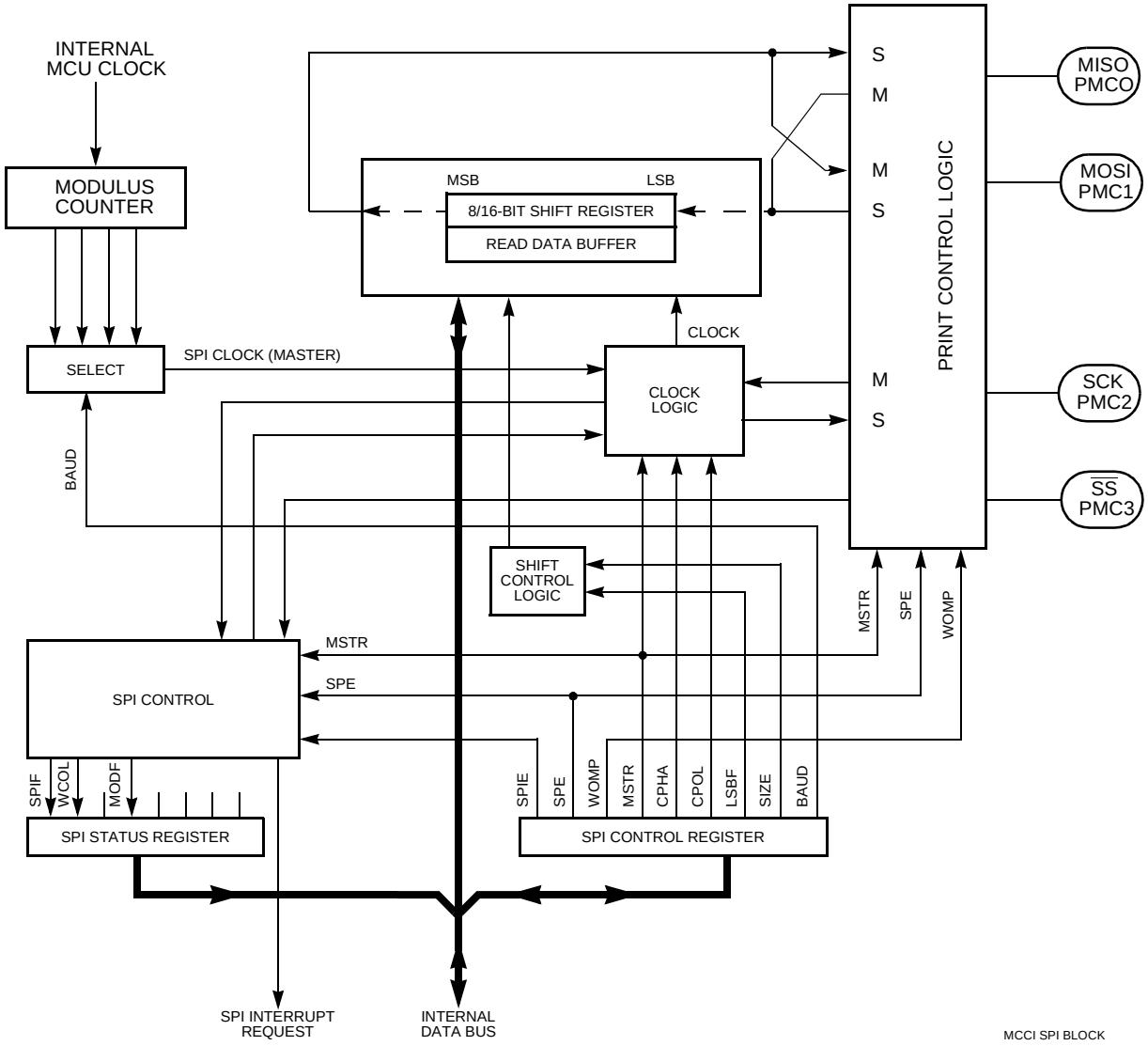


Figure 11-2 SPI Block Diagram

Clock control logic allows a selection of clock polarity and a choice of two clocking protocols to accommodate most available synchronous serial peripheral devices. When the SPI is configured as a master, software selects one of 254 different bit rates for the serial clock.

During an SPI transfer, data is simultaneously transmitted (shifted out serially) and received (shifted in serially). A serial clock line synchronizes shifting and sampling of the information on the two serial data lines. A slave-select line allows individual selection of a slave SPI device. Slave devices which are not selected do not interfere with SPI bus activities. On a master SPI device the slave-select line can optionally be used to indicate a multiple-master bus contention.

Error-detection logic is included to support interprocessor interfacing. A write-collision detector indicates when an attempt is made to write data to the serial shift register while a transfer is in progress. A multiple-master mode-fault detector automatically disables SPI output drivers if more than one MCU simultaneously attempts to become bus master.

11.3.1 SPI Registers

SPI control registers include the SPI control register (SPCR), the SPI status register (SPSR), and the SPI data register (SPDR). Refer to **APPENDIX D REGISTER SUMMARY** for register bit and field definitions.

11.3.1.1 SPI Control Register (SPCR)

The SPCR contains parameters for configuring the SPI. The register can be read or written at any time.

11.3.1.2 SPI Status Register (SPSR)

The SPSR contains SPI status information. Only the SPI can set the bits in this register. The CPU reads the register to obtain status information.

11.3.1.3 SPI Data Register (SPDR)

The SPDR is used to transmit and receive data on the serial bus. A write to this register in the master device initiates transmission or reception of another byte or word. After a byte or word of data is transmitted, the SPIF status bit is set in both the master and slave devices.

A read of the SPDR actually reads a buffer. If the first SPIF is not cleared by the time a second transfer of data from the shift register to the read buffer is initiated, an overrun condition occurs. In cases of overrun the byte or word causing the overrun is lost.

A write to the SPDR is not buffered and places data directly into the shift register for transmission.

11.3.2 SPI Pins

Four bi-directional pins are associated with the SPI. The MPAR configures each pin for either SPI function or general-purpose I/O. The MDDR assigns each pin as either input or output. The WOMP bit in the SPI control register (SPCR) determines whether each SPI pin that is configured for output functions as an open-drain output or a normal CMOS output. The MDDR and WOMP assignments are valid regardless of whether the pins are configured for SPI use or general-purpose I/O.

The operation of pins configured for SCI use depends on whether the SCI is operating as a master or a slave, determined by the MSTR bit in the SPCR.

Table 11-3 shows SPI pins and their functions.

Table 11-3 SPI Pin Functions

Pin Name	Mode	Function
Master in, slave out (MISO)	Master	Provides serial data input to the SPI
	Slave	Provides serial data output from the SPI
Master out, slave in (MOSI)	Master	Provides serial output from the SPI
	Slave	Provides serial input to the SPI
Serial clock (SCK)	Master	Provides clock output from the SPI
	Slave	Provides clock input to the SPI
Slave select ($\overline{SS}$)	Master	Detects bus-master mode fault
	Slave	Selects the SPI for an externally-initiated serial transfer

11.3.3 SPI Operating Modes

The SPI operates in either master or slave mode. Master mode is used when the MCU originates data transfers. Slave mode is used when an external device initiates serial transfers to the MCU. The MSTR bit in SPCR selects master or slave operation.

11.3.3.1 Master Mode

Setting the MSTR bit in SPCR selects master mode operation. In master mode, the SPI can initiate serial transfers but cannot respond to externally initiated transfers. When the slave-select input of a device configured for master mode is asserted, a mode fault occurs.

When using the SPI in master mode, include the following steps:

1. Write to the MMCR, MIVR, and ILSPI. Refer to **11.5 MCCI Initialization** for more information.
2. Write to the MPAR to assign the following pins to the SPI: MISO, MOSI, and (optionally) $\overline{SS}$. MISO is used for serial data input in master mode, and MOSI is used for serial data output. Either or both may be necessary, depending on the particular application. $\overline{SS}$ is used to generate a mode fault in master mode. If this SPI is the only possible master in the system, the $\overline{SS}$ pin may be used for general-purpose I/O.
3. Write to the MDDR to direct the data flow on SPI pins. Configure the SCK (serial clock) and MOSI pins as outputs. Configure MISO and (optionally) $\overline{SS}$ as inputs.
4. Write to the SPCR to assign values for BAUD, CPHA, CPOL, SIZE, LSBF, WOMP, and SPIE. Set the MSTR bit to select master operation. Set the SPE bit to enable the SPI.
5. Enable the slave device.
6. Write appropriate data to the SPI data register to initiate the transfer.

When the SPI reaches the end of the transmission, it sets the SPIF flag in the SPSR. If the SPIE bit in the SPCR is set, an interrupt request is generated when SPIF is asserted. After the SPSR is read with SPIF set, and then the SPDR is read or written to, the SPIF flag is automatically cleared.

Data transfer is synchronized with the internally-generated serial clock (SCK). Control bits CPHA and CPOL in SPCR control clock phase and polarity. Combinations of CPHA and CPOL determine the SCK edge on which the master MCU drives outgoing data from the MOSI pin and latches incoming data from the MISO pin.

11.3.3.2 Slave Mode

Clearing the MSTR bit in SPCR selects slave mode operation. In slave mode, the SPI is unable to initiate serial transfers. Transfers are initiated by an external bus master. Slave mode is typically used on a multimaster SPI bus. Only one device can be bus master (operate in master mode) at any given time.

When using the SPI in slave mode, include the following steps:

1. Write to the MMCR and interrupt registers. Refer to **11.5 MCCI Initialization** for more information.
2. Write to the MPAR to assign the following pins to the SPI: MISO, MOSI, and $\overline{SS}$. MISO is used for serial data output in slave mode, and MOSI is used for serial data input. Either or both may be necessary, depending on the particular application. SCK is the input serial clock. $\overline{SS}$ selects the SPI when asserted.
3. Write to the MDDR to direct the data flow on SPI pins. Configure the SCK, MOSI, and $\overline{SS}$ pins as inputs. Configure MISO as an output.
4. Write to the SPCR to assign values for CPHA, CPOL, SIZE, LSBF, WOMP, and SPIE. Set the MSTR bit to select master operation. Set the SPE bit to enable the SPI. (The BAUD field in the SPCR of the slave device has no effect on SPI operation.)

When SPE is set and MSTR is clear, a low state on the $\overline{SS}$ pin initiates slave mode operation. The $\overline{SS}$ pin is used only as an input.

After a byte or word of data is transmitted, the SPI sets the SPIF flag. If the SPIE bit in SPCR is set, an interrupt request is generated when SPIF is asserted.

Transfer is synchronized with the externally generated SCK. The CPHA and CPOL bits determine the SCK edge on which the slave MCU latches incoming data from the MOSI pin and drives outgoing data from the MISO pin.

11.3.4 SPI Clock Phase and Polarity Controls

Two bits in the SPCR determine SCK phase and polarity. The clock polarity (CPOL) bit selects clock polarity (high true or low true clock). The clock phase control bit (CPHA) selects one of two transfer formats and affects the timing of the transfer. The clock phase and polarity should be the same for the master and slave devices. In some cases, the phase and polarity may be changed between transfers to allow a master device to communicate with slave devices with different requirements. The flexibility of the SPI system allows it to be directly interfaced to almost any existing synchronous serial peripheral.

11.3.4.1 CPHA = 0 Transfer Format

Figure 11-3 is a timing diagram of an eight-bit, MSB-first SPI transfer in which CPHA equals zero. Two waveforms are shown for SCK: one for CPOL equal to zero and another for CPOL equal to one. The diagram may be interpreted as a master or slave timing diagram since the SCK, MISO and MOSI pins are directly connected between the master and the slave. The MISO signal shown is the output from the slave and the MOSI signal shown is the output from the master. The $\overline{SS}$ line is the chip-select input to the slave.

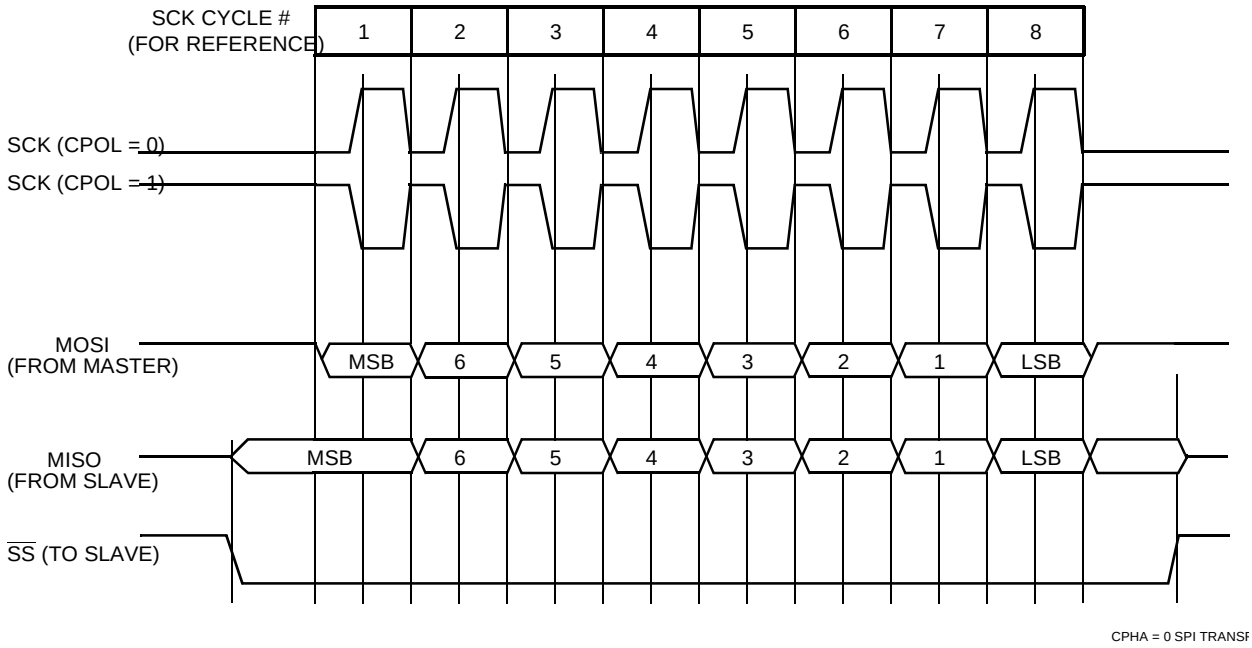


Figure 11-3 CPHA = 0 SPI Transfer Format

For a master, writing to the SPDR initiates the transfer. For a slave, the falling edge of $\overline{SS}$ indicates the start of a transfer. The SCK signal remains inactive for the first half of the first SCK cycle. Data is latched on the first and each succeeding odd clock edge, and the SPI shift register is left-shifted on the second and succeeding even clock edges. SPIF is set at the end of the eighth SCK cycle.

When CPHA equals zero, the $\overline{SS}$ line must be negated and reasserted between each successive serial byte. If the slave writes data to the SPI data register while $\overline{SS}$ is asserted (low), a write collision error results. To avoid this problem, the slave should read bit three of PORTMCP, which indicates the state of the $\overline{SS}$ pin, before writing to the SPDR again.

11.3.4.2 CPHA = 1 Transfer Format

Figure 11-4 is a timing diagram of an eight-bit, MSB-first SPI transfer in which CPHA equals one. Two waveforms are shown for SCK, one for CPOL equal to zero and another for CPOL equal to one. The diagram may be interpreted as a master or slave timing diagram since the SCK, MISO and MOSI pins are directly connected between the master and the slave. The MISO signal shown is the output from the slave and the MOSI signal shown is the output from the master. The $\overline{SS}$ line is the slave select input to the slave.

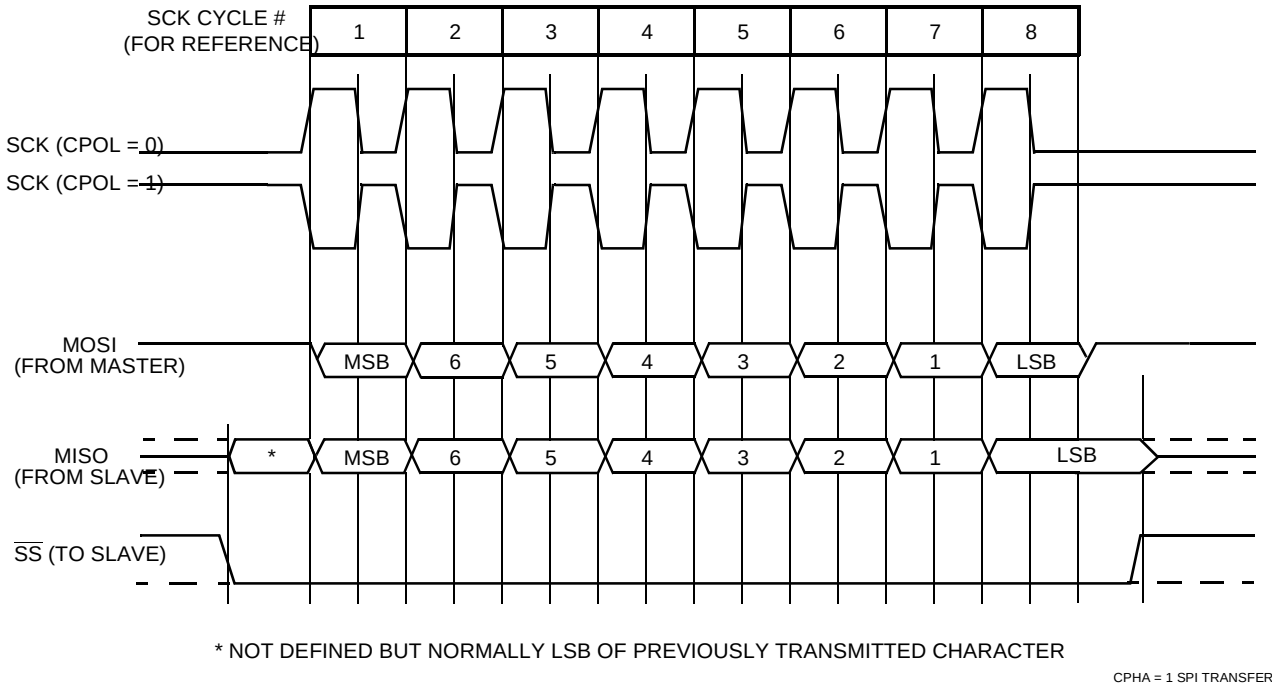


Figure 11-4 CPHA = 1 SPI Transfer Format

For a master, writing to the SPDR initiates the transfer. For a slave, the first edge of SCK indicates the start of a transfer. The SPI is left-shifted on the first and each succeeding odd clock edge, and data is latched on the second and succeeding even clock edges.

SCK is inactive for the last half of the eighth SCK cycle. For a master, SPIF is set at the end of the eighth SCK cycle (after the seventeenth SCK edge). Since the last SCK edge occurs in the middle of the eighth SCK cycle, however, the slave has no way of knowing when the end of the last SCK cycle occurs. The slave therefore considers the transfer complete after the last bit of serial data has been sampled, which corresponds to the middle of the eighth SCK cycle.

When CPHA is one, the $\overline{SS}$ line may remain at its active low level between transfers. This format is sometimes preferred in systems having a single fixed master and only one slave that needs to drive the MISO data line.

11.3.5 SPI Serial Clock Baud Rate

Baud rate is selected by writing a value from 2 to 255 into SPBR[7:0] in the SPCR of the master MCU. Writing a SPBR[7:0] value into the SPCR of the slave device has no effect. The SPI uses a modulus counter to derive SCK baud rate from the MCU system clock.

The following expressions apply to SCK baud rate:

$$\text{SCK Baud Rate} = \frac{f_{sys}}{2 \times \text{SPBR[7:0]}}$$

or

$$\text{SPBR[7:0]} = \frac{f_{sys}}{2 \times \text{SCK Baud Rate Desired}}$$

Giving SPBR[7:0] a value of zero or one disables the baud rate generator. SCK is disabled and assumes its inactive state value.

SPBR[7:0] has 254 active values. **Table 11-4** lists several possible baud values and the corresponding SCK frequency based on a 16.78-MHz system clock.

Table 11-4 SCK Frequencies

System Clock Frequency	Required Division Ratio	Value of SPBR	Actual SCK Frequency
16.78 MHz	4	2	4.19 MHz
	8	4	2.10 MHz
	16	8	1.05 MHz
	34	17	493 kHz
	168	84	100 kHz
	510	255	33 kHz

11.3.6 Wired-OR Open-Drain Outputs

Typically, SPI bus outputs are not open-drain unless multiple SPI masters are in the system. If needed, the WOMP bit in SPCR can be set to provide wired-OR, open-drain outputs. An external pull-up resistor should be used on each output line. WOMP affects all SPI pins regardless of whether they are assigned to the SPI or used as general-purpose I/O.

11.3.7 Transfer Size and Direction

The SIZE bit in the SPCR selects a transfer size of 8 (SIZE = 0) or 16 (SIZE = 1) bits. The LSBF bit in the SPCR determines whether serial shifting to and from the data register begins with the LSB (LSBF = 1) or MSB (LSBF = 0).

11.3.8 Write Collision

A write collision occurs if an attempt is made to write the SPDR while a transfer is in progress. Since the SPDR is not double buffered in the transmit direction, a successful write to SPDR would cause data to be written directly into the SPI shift register. Because this would corrupt any transfer in progress, a write collision error is generated instead. The transfer continues undisturbed, the data that caused the error is not written to the shifter, and the WCOL bit in SPSR is set. No SPI interrupt is generated.

A write collision is normally a slave error because a slave has no control over when a master initiates a transfer. Since a master is in control of the transfer, software can avoid a write collision error generated by the master. The SPI logic can, however, detect a write collision in a master as well as in a slave.

What constitutes a transfer in progress depends on the SPI configuration. For a master, a transfer starts when data is written to the SPDR and ends when SPIF is set. For a slave, the beginning and ending points of a transfer depend on the value of CPHA. When CPHA = 0, the transfer begins when $\overline{SS}$ is asserted and ends when it is negated. When CPHA = 1, a transfer begins at the edge of the first SCK cycle and ends when SPIF is set. Refer to **11.3.4 SPI Clock Phase and Polarity Controls** for more information on transfer periods and on avoiding write collision errors.

When a write collision occurs, the WCOL bit in the SPSR is set. To clear WCOL, read the SPSR while WCOL is set, and then either read the SPDR (either before or after SPIF is set) or write the SPDR after SPIF is set. (Writing the SPDR before SPIF is set results in a second write collision error.) This process clears SPIF as well as WCOL.

11.3.9 Mode Fault

When the SPI system is configured as a master and the $\overline{SS}$ input line is asserted, a mode fault error occurs, and the MODF bit in the SPSR is set. Only an SPI master can experience a mode fault error, caused when a second SPI device becomes a master and selects this device as if it were a slave.

To avoid latchup caused by contention between two pin drivers, the MCU does the following when it detects a mode fault error:

1. Forces the MSTR control bit to zero to reconfigure the SPI as a slave.
2. Forces the SPE control bit to zero to disable the SPI system.
3. Sets the MODF status flag and generates an SPI interrupt if SPIE = 1.
4. Clears the appropriate bits in the MDDR to configure all SPI pins except the $\overline{SS}$ pin as inputs.

After correcting the problems that led to the mode fault, clear MODF by reading the SPSR while MODF is set and then writing to the SPCR. Control bits SPE and MSTR may be restored to their original set state during this clearing sequence or after the MODF bit has been cleared. Hardware does not allow the user to set the SPE and MSTR bits while MODF is a logic one except during the proper clearing sequence.

11.4 Serial Communication Interface (SCI)

The SCI submodule contains two independent SCI systems. Each is a full-duplex universal asynchronous receiver transmitter (UART). This SCI system is fully compatible with SCI systems found on other Motorola devices, such as the M68HC11 and M68HC05 families.

The SCI uses a standard non-return to zero (NRZ) transmission format. An on-chip baud-rate generator derives standard baud-rate frequencies from the MCU oscillator. Both the transmitter and the receiver are double buffered, so that back-to-back characters can be handled easily even if the CPU is delayed in responding to the completion of an individual character. The SCI transmitter and receiver are functionally independent but use the same data format and baud rate.

Figure 11-5 shows a block diagram of the SCI transmitter. **Figure 11-6** shows a block diagram of the SCI receiver.

The two independent SCI systems are called SCIA and SCIB. These SCIs are identical in register set and hardware configuration, providing an application with full flexibility in using the dual SCI system. References to SCI registers in this section do not always distinguish between the two SCI systems. A reference to SCCR1, for example, applies to both SCCR1A (SCIA control register 1) and SCCR1B (SCIB control register 1).

11.4.1 SCI Registers

The SCI programming model includes the MCCI global and pin control registers and eight SCI registers. Each of the two SCI units contains two SCI control registers, one status register, and one data register. Refer to **APPENDIX D REGISTER SUMMARY** for register bit and field definitions.

All registers may be read or written at any time by the CPU. Rewriting the same value to any SCI register does not disrupt operation; however, writing a different value into an SCI register when the SCI is running may disrupt operation. To change register values, the receiver and transmitter should be disabled with the transmitter allowed to finish first. The status flags in the SCSR may be cleared at any time.

When initializing the SCI, set the transmitter enable (TE) and receiver enable (RE) bits in SCCR1 last. A single word write to SCCR1 can be used to initialize the SCI and enable the transmitter and receiver.

11.4.1.1 SCI Control Registers

SCCR0 contains the baud rate selection field. The baud rate must be set before the SCI is enabled. The CPU16 can read and write this register at any time.

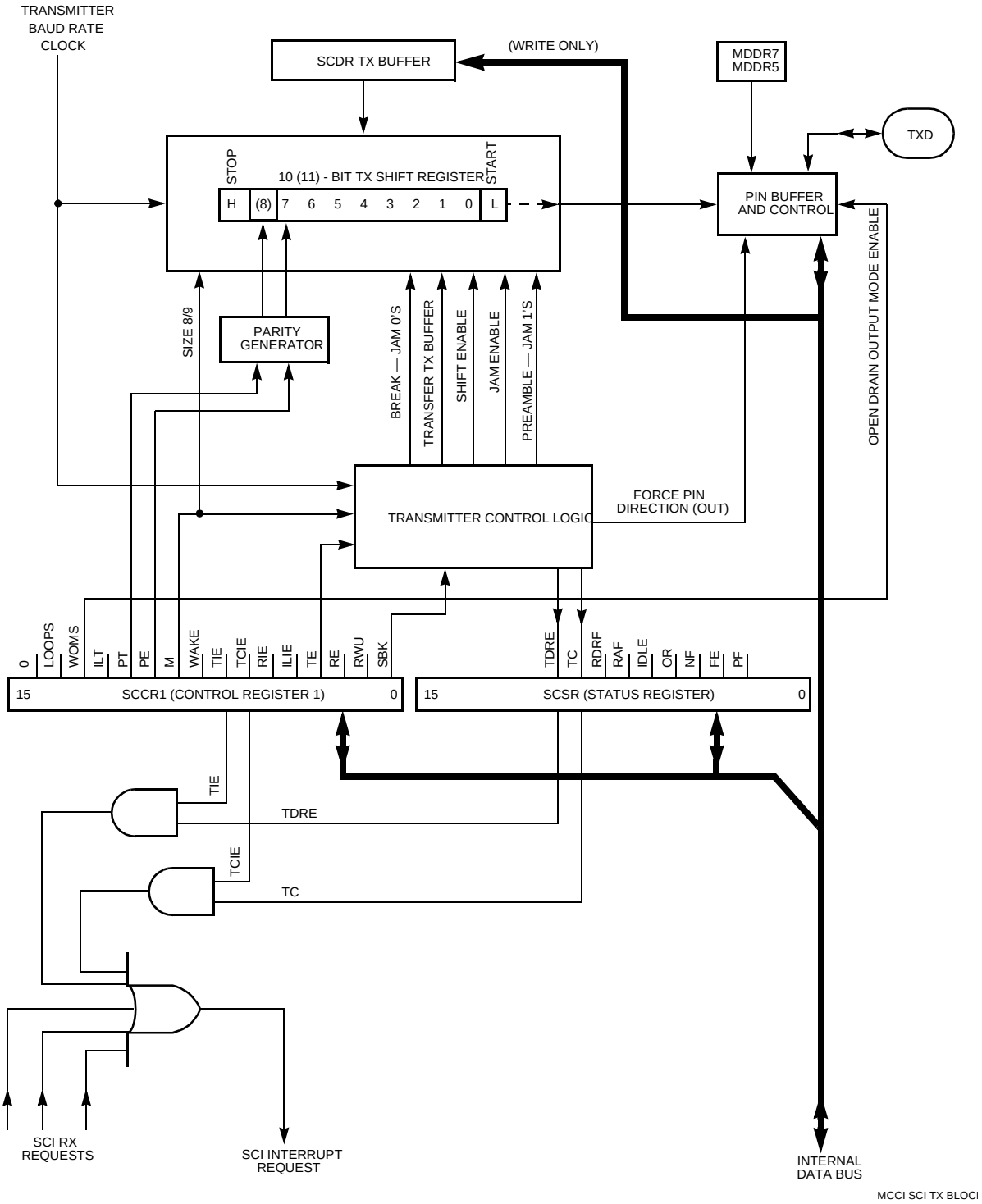


Figure 11-5 SCI Transmitter Block Diagram

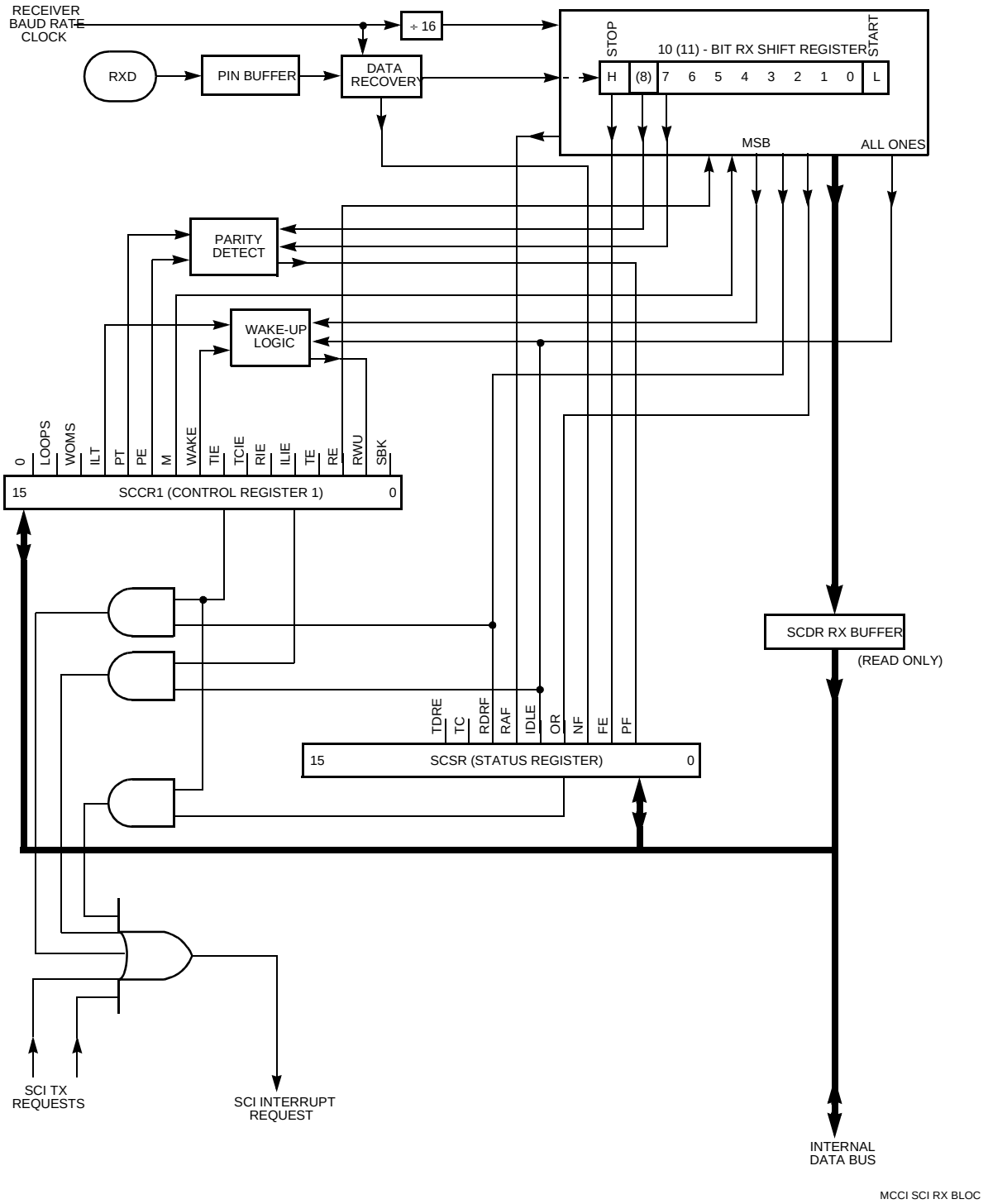


Figure 11-6 SCI Receiver Block Diagram

SCCR1 contains a number of SCI configuration parameters, including transmitter and receiver enable bits, interrupt enable bits, and operating mode enable bits. The CPU16 can read and write this register at any time. The SCI can modify the RWU bit under certain circumstances.

Changing the value of SCI control bits during a transfer may disrupt operation. Before changing register values, allow the SCI to complete the current transfer, then disable the receiver and transmitter.

11.4.1.2 SCI Status Register

The SCSR contains flags that show SCI operating conditions. These flags are cleared either by SCI hardware or by a read/write sequence. To clear SCI transmitter flags, read the SCSR and then write to the SCDR. To clear SCI receiver flags, read the SCSR and then read the SCDR. A long-word read can consecutively access both the SCSR and the SCDR. This action clears receiver status flag bits that were set at the time of the read, but does not clear TDRE or TC flags.

If an internal SCI signal for setting a status bit comes after the CPU has read the asserted status bits, but before the CPU has written or read the SCDR, the newly set status bit is not cleared. The SCSR must be read again with the bit set, and the SCDR must be written to or read before the status bit is cleared.

Reading either byte of the SCSR causes all 16 bits to be accessed, and any status bit already set in either byte will be cleared on a subsequent read or write of the SCDR.

11.4.1.3 SCI Data Register

The SCDR contains two data registers at the same address. The RDR is a read-only register that contains data received by the SCI serial interface. The data comes into the receive serial shifter and is transferred to the RDR. The TDR is a write-only register that contains data to be transmitted. The data is first written to the TDR, then transferred to the transmit serial shifter, where additional format bits are added before transmission.

11.4.2 SCI Pins

Four pins are associated with the SCI: TXDA, TXDB, RXDA, and RXDB. The state of the TE or RE bit in SCI control register 1 of each SCI submodule (SCCR1A, SCCR1B) determines whether the associated pin is configured for SCI operation or general-purpose I/O. The MDDR assigns each pin as either input or output. The WOMC bit in SCCR1A or SCCR1B determines whether the associated RXD and TXD pins, when configured as outputs, function as open-drain output pins or normal CMOS outputs. The MDDR and WOMC assignments are valid regardless of whether the pins are configured for SPI use or general-purpose I/O.

SCI pins are listed in **Table 11-5**.

Table 11-5 SCI Pins

Pin	Mode	SCI Function	Port I/O Signal
Transmit data	TXDA	Serial data output from SCIA (TE = 1)	PMC7
	TXDB	Serial data output from SCIB (TE = 1)	PMC5
Receive data	RXDA	Serial data input to SCIA (RE = 1)	PMC6
	RXDB	Serial data input to SCIB (RE = 1)	PMC4

11.4.3 Receive Data Pins (RXDA, RXDB)

RXDA and RXDB are the serial data inputs to the SCIA and SCIB interfaces, respectively. Each pin is also available as a general-purpose I/O pin when the RE bit in SCCR1 of the associated SCI submodule is cleared. When used for general-purpose I/O, RXDA and RXDB may be configured either as input or output as determined by the RXDA and RXDB bits in the MDDR.

11.4.4 Transmit Data Pins (TXDA, TXDB)

When used for general-purpose I/O, TXDA and TXDB can be configured either as input or output as determined by the TXDA and TXDB bits in the MDDR. The TXDA and TXDB pins are enabled for SCI use by setting the TE bit in SCCR1 of each SCI interface.

11.4.5 SCI Operation

SCI operation can be polled by means of status flags in the SCSR, or interrupt-driven operation can be employed by means of the interrupt-enable bits in SCCR1.

11.4.5.1 Definition of Terms

Data can be transmitted and received in a number of formats. The following terms concerning data format are used in this section:

- Bit-Time — The time required to transmit or receive one bit of data, which is equal to one cycle of the baud frequency.
- Start Bit — One bit-time of logic zero that indicates the beginning of a data frame. A start bit must begin with a one-to-zero transition and be preceded by at least three receive time samples of logic one.
- Stop Bit— One bit-time of logic one that indicates the end of a data frame.
- Frame — A complete unit of serial information. The SCI can use 10-bit or 11-bit frames.
- Data Frame — A start bit, a specified number of data or information bits, and at least one stop bit.
- Idle Frame — A frame that consists of consecutive ones. An idle frame has no start bit.
- Break Frame — A frame that consists of consecutive zeros. A break frame has no stop bits.

11.4.5.2 Serial Formats

All data frames must have a start bit and at least one stop bit. Receiving and transmitting devices must use the same data frame format. The SCI provides hardware support for both 10-bit and 11-bit frames. The M bit in SCCR1 specifies the number of bits per frame.

The most common data frame format for NRZ serial interfaces is one start bit, eight data bits (LSB first), and one stop bit; a total of ten bits. The most common 11-bit data frame contains one start bit, eight data bits, a parity or control bit, and one stop bit. Ten-bit and eleven-bit frames are shown in **Table 11-6**.

Table 11-6 Serial Frame Formats

10-Bit Frames			
Start	**Data**	**Parity/Control**	**Stop**
1	7	—	2
1	7	1	1
1	8	—	1
11-Bit Frames			
Start	**Data**	**Parity/Control**	**Stop**
1	7	1	2
1	8	1	1

11.4.5.3 Baud Clock

The SCI baud rate is programmed by writing a 13-bit value to the SCBR field in SCI control register zero (SCCR0). The baud rate is derived from the MCU system clock by a modulus counter. Writing a value of zero to SCBR[12:0] disables the baud rate generator. Baud rate is calculated as follows:

$$\text{SCI Baud Rate} = \frac{f_{sys}}{32 \times \text{SCBR[12:0]}}$$

or

$$\text{SCBR[12:0]} = \frac{f_{sys}}{32 \times \text{SCI Baud Rate Desired}}$$

where SCBR[12:0] is in the range {1, 2, 3, ..., 8191}.

The SCI receiver operates asynchronously. An internal clock is necessary to synchronize with an incoming data stream. The SCI baud rate generator produces a receive time sampling clock with a frequency 16 times that of the SCI baud rate. The SCI determines the position of bit boundaries from transitions within the received waveform, and adjusts sampling points to the proper positions within the bit period.

11.4.5.4 Parity Checking

The PT bit in SCCR1 selects either even (PT = 0) or odd (PT = 1) parity. PT affects received and transmitted data. The PE bit in SCCR1 determines whether parity checking is enabled (PE = 1) or disabled (PE = 0). When PE is set, the MSB of data in a frame is used for the parity function. For transmitted data, a parity bit is generated for received data; the parity bit is checked. When parity checking is enabled, the PF bit in the SCI status register (SCSR) is set if a parity error is detected.

Enabling parity affects the number of data bits in a frame, which can in turn affect frame size. **Table 11-7** shows possible data and parity formats.

Table 11-7 Effect of Parity Checking on Data Size

M	PE	Result
0	0	8 data bits
0	1	7 data bits, 1 parity bit
1	0	9 data bits
1	1	8 data bits, 1 parity bit

11.4.5.5 Transmitter Operation

The transmitter consists of a serial shifter and a parallel data register (TDR) located in the SCI data register (SCDR). The serial shifter cannot be directly accessed by the CPU16. The transmitter is double-buffered, which means that data can be loaded into the TDR while other data is shifted out. The TE bit in SCCR1 enables (TE = 1) and disables (TE = 0) the transmitter.

Shifter output is connected to the TXD pin while the transmitter is operating (TE = 1, or TE = 0 and transmission in progress). Wired-OR operation should be specified when more than one transmitter is used on the same SCI bus. The WOMS bit in SCCR1 determines whether TXD is an open-drain (wired-OR) output or a normal CMOS output. An external pull-up resistor on TXD is necessary for wired-OR operation. WOMS controls TXD function whether the pin is used by the SCI or as a general-purpose I/O pin.

Data to be transmitted is written to SCDR, then transferred to the serial shifter. The transmit data register empty (TDRE) flag in SCSR shows the status of TDR. When TDRE = 0, the TDR contains data that has not been transferred to the shifter. Writing to SCDR again overwrites the data. TDRE is set when the data in the TDR is transferred to the shifter. Before new data can be written to the SCDR, however, the processor must clear TDRE by writing to SCSR. If new data is written to the SCDR without first clearing TDRE, the data will not be transmitted.

The transmission complete (TC) flag in SCSR shows transmitter shifter state. When TC = 0, the shifter is busy. TC is set when all shifting operations are completed. TC is not automatically cleared. The processor must clear it by first reading SCSR while TC is set, then writing new data to SCDR.

The state of the serial shifter is checked when the TE bit is set. If TC = 1, an idle frame is transmitted as a preamble to the following data frame. If TC = 0, the current operation continues until the final bit in the frame is sent, then the preamble is transmitted. The TC bit is set at the end of preamble transmission.

The SBK bit in SCCR1 is used to insert break frames in a transmission. A non-zero integer number of break frames is transmitted while SBK is set. Break transmission begins when SBK is set, and ends with the transmission in progress at the time either SBK or TE is cleared. If SBK is set while a transmission is in progress, that transmission finishes normally before the break begins. To assure the minimum break time, toggle SBK quickly to one and back to zero. The TC bit is set at the end of break transmission. After break transmission, at least one bit-time of logic level one (mark idle) is transmitted to ensure that a subsequent start bit can be detected.

If TE remains set, after all pending idle, data and break frames are shifted out, TDRE and TC are set and TXD is held at logic level one (mark).

When TE is cleared, the transmitter is disabled after all pending idle; data and break frames are transmitted. The TC flag is set, and control of the TXD pin reverts to PQSPAR and DDRQS. Buffered data is not transmitted after TE is cleared. To avoid losing data in the buffer, do not clear TE until TDRE is set.

Some serial communication systems require a mark on the TXD pin even when the transmitter is disabled. Configure the TXD pin as an output, then write a one to PQS7. When the transmitter releases control of the TXD pin, it reverts to driving a logic one output.

To insert a delimiter between two messages, to place non-listening receivers in wake-up mode between transmissions, or to signal a retransmission by forcing an idle line, clear and then set TE before data in the serial shifter has shifted out. The transmitter finishes the transmission, then sends a preamble. After the preamble is transmitted, if TDRE is set, the transmitter will mark idle. Otherwise, normal transmission of the next sequence will begin.

Both TDRE and TC have associated interrupts. The interrupts are enabled by the transmit interrupt enable (TIE) and transmission complete interrupt enable (TCIE) bits in SCCR1. Service routines can load the last byte of data in a sequence into SCDR, then terminate the transmission when a TDRE interrupt occurs.

11.4.5.6 Receiver Operation

The RE bit in SCCR1 enables (RE = 1) and disables (RE = 0) the receiver. The receiver contains a receive serial shifter and a parallel receive data register (RDR) located in the SCI data register (SCDR). The serial shifter cannot be directly accessed by the CPU16. The receiver is double-buffered, allowing data to be held in the RDR while other data is shifted in.

Receiver bit processor logic drives a state machine that determines the logic level for each bit-time. This state machine controls when the bit processor logic is to sample the RXD pin and also controls when data is to be passed to the receive serial shifter.

A receive time clock is used to control sampling and synchronization. Data is shifted into the receive serial shifter according to the most recent synchronization of the receive time clock with the incoming data stream. From this point on, data movement is synchronized with the MCU system clock.

The number of bits shifted in by the receiver depends on the serial format. However, all frames must end with at least one stop bit. When the stop bit is received, the frame is considered to be complete, and the received data in the serial shifter is transferred to the RDR. The receiver data register flag (RDRF) is set when the data is transferred.

Noise errors, parity errors, and framing errors can be detected while a data stream is being received. Although error conditions are detected as bits are received, the noise flag (NF), the parity flag (PF), and the framing error (FE) flag in SCSR are not set until data is transferred from the serial shifter to the RDR.

RDRF must be cleared before the next transfer from the shifter can take place. If RDRF is set when the shifter is full, transfers are inhibited and the overrun error (OR) flag in SCSR is set. OR indicates that the RDR needs to be serviced faster. When OR is set, the data in the RDR is preserved, but the data in the serial shifter is lost. Because framing, noise, and parity errors are detected while data is in the serial shifter, FE, NF, and PF cannot occur at the same time as OR.

When the CPU16 reads SCSR and SCDR in sequence, it acquires status and data, and also clears the status flags. Reading SCSR acquires status and arms the clearing mechanism. Reading SCDR acquires data and clears SCSR.

When RIE in SCCR1 is set, an interrupt request is generated whenever RDRF is set. Because receiver status flags are set at the same time as RDRF, they do not have separate interrupt enables.

11.4.5.7 Idle-Line Detection

During a typical serial transmission, frames are transmitted isochronally and no idle time occurs between frames. Even when all the data bits in a frame are logic ones, the start bit provides one logic zero bit-time during the frame. An idle line is a sequence of contiguous ones equal to the current frame size. Frame size is determined by the state of the M bit in SCCR1.

The SCI receiver has both short and long idle-line detection capability. Idle-line detection is always enabled. The idle line type (ILT) bit in SCCR1 determines which type of detection is used. When an idle line condition is detected, the IDLE flag in SCSR is set.

For short idle-line detection, the receiver bit processor counts contiguous logic one bit-times whenever they occur. Short detection provides the earliest possible recognition of an idle line condition, because the stop bit and contiguous logic ones before and after it are counted. For long idle-line detection, the receiver counts logic ones after the stop bit is received. Only a complete idle frame causes the IDLE flag to be set.

In some applications, software overhead can cause a bit-time of logic level one to occur between frames. This bit-time does not affect content, but if it occurs after a frame of ones when short detection is enabled, the receiver flags an idle line.

When the ILIE bit in SCCR1 is set, an interrupt request is generated when the IDLE flag is set. The flag is cleared by reading SCSR and SCDR in sequence. IDLE is not set again until after at least one frame has been received (RDRF = 1). This prevents an extended idle interval from causing more than one interrupt.

11.4.5.8 Receiver Wake-Up

The receiver wake-up function allows a transmitting device to direct a transmission to a single receiver or to a group of receivers by sending an address frame at the start of a message. Hardware activates each receiver in a system under certain conditions. Resident software must process address information and enable or disable receiver operation.

A receiver is placed in wake-up mode by setting the RWU bit in SCCR1. While RWU is set, receiver status flags and interrupts are disabled. Although the CPU32 can clear RWU, it is normally cleared by hardware during wake-up.

The WAKE bit in SCCR1 determines which type of wake-up is used. When WAKE = 0, idle-line wake-up is selected. When WAKE = 1, address-mark wake-up is selected. Both types require a software-based device addressing and recognition scheme.

Idle-line wake-up allows a receiver to sleep until an idle line is detected. When an idle-line is detected, the receiver clears RWU and wakes up. The receiver waits for the first frame of the next transmission. The byte is received normally, transferred to the RDR, and the RDRF flag is set. If software does not recognize the address, it can set RWU and put the receiver back to sleep. For idle-line wake-up to work, there must be a minimum of one frame of idle line between transmissions. There must be no idle time between frames within a transmission.

Address-mark wake-up uses a special frame format to wake up the receiver. When the MSB of an address-mark frame is set, that frame contains address information. The first frame of each transmission must be an address frame. When the MSB of a frame is set, the receiver clears RWU and wakes up. The byte is received normally, transferred to the RDR, and the RDRF flag is set. If software does not recognize the address, it can set RWU and put the receiver back to sleep. Address-mark wake-up allows idle time between frames and eliminates idle time between transmissions. However, there is a loss of efficiency because of an additional bit-time per frame.

11.4.5.9 Internal Loop

The LOOPS bit in SCCR1 controls a feedback path in the data serial shifter. When LOOPS is set, the SCI transmitter output is fed back into the receive serial shifter. TXD is asserted (idle line). Both transmitter and receiver must be enabled before entering loop mode.

11.5 MCCI Initialization

After reset, the MCCI remains in an idle state. Several registers must be initialized before serial operations begin. A general sequence guide for initialization follows.

1. Global
   a. Configure MMCR
      1. Write an interrupt arbitration number greater than zero into the IARB field.
      2. Clear the STOP bit if it is not already cleared.
   b. Interrupt vector and interrupt level registers (MIVR, ILSPI, and ILSCI)
      1. Write the SPI/SCI interrupt vector into MIVR.
      2. Write the SPI interrupt request level into the ILSPI and the interrupt request levels for the two SCI interfaces into the ILSCI.
   c. Port data register
      1. Write a data word to PORTMC.
      2. Read a port pin state from PORTMCP.
   d. Pin control registers
      1. Establish the direction of MCCI pins by writing to the MDDR.
      2. Assign pin functions by writing to the MPAR.
2. Serial Peripheral Interface
   a. Configure SPCR
      1. Write a transfer rate value into the BAUD field.
      2. Determine clock phase (CPHA) and clock polarity (CPOL).
      3. Specify an 8- or 16-bit transfer (SIZE) and MSB- or LSB-first transfer mode (LSBF).
      4. Select master or slave operating mode (MSTR).
      5. Enable or disable wired-OR operation (WOMP).
      6. Enable or disable SPI interrupts (SPIE).
      7. Enable the SPI by setting the SPE bit.
3. Serial Communication Interface (SCIA/SCIB)
   a. To transmit, read the SCSR, and then write transmit data to the SCDR. This clears the TDRE and TC indicators in the SCSR.
      1. SCI control register 0 (SCCR0)
      2. Write a baud rate value into the BR field.
   b. Configure SCCR1
      1. Select 8- or 9-bit frame format (M).
      2. Determine use (PE) and type (PT) of parity generation or detection.
      3. To receive, set the RE and RIE bits in SCCR1. Select use (RWU) and type (WAKE) of receiver wakeup. Select idle-line detection type (ILT) and enable or disable idle-line interrupt (ILIE).
      4. To transmit, set TE and TIE bits in SCCR1, and enable or disable WOMC and TCIE bits. Disable break transmission (SBK) for normal operation.

SECTION 12
CONFIGURABLE TIMER MODULE

This section is an overview of the configurable timer module (CTM). Refer to the *CTM Reference Manual* (CTMRM/AD) for a comprehensive discussion of CTM capabilities.

12.1 General

MC68HC16R1 and MC68HC916R1 MCUs use the CTM7. MC68HC16R3 and MC68HC916R3 MCUs use the CTM8.

NOTE

The CTM7 and the CTM8 are referred to as “CTM” except when specific differences require separate identification.

The CTM consists of several submodules which are located on either side of the CTM internal submodule bus (SMB). All data and control signals within the CTM are passed over this bus. The SMB is connected to the outside world via the bus interface unit submodule (BIUSM), which is connected to the intermodule bus (IMB), and subsequently the CPU16. This configuration allows the CPU16 to access the data and control registers in each CTM submodule on the SMB. The MC68HC16R1 and the MC68HC916R1 use two time base buses (TBB1 and TBB2), each 16-bits wide, to transfer timing information from counters to action submodules. The MC68HC16R3 and the MC68HC916R3 use two additional time base buses (TBB3 and TBB4), each 16-bits wide.

Figure 12-1 shows a block diagram of the CTM7. **Figure 12-2** shows a block diagram of the CTM8.

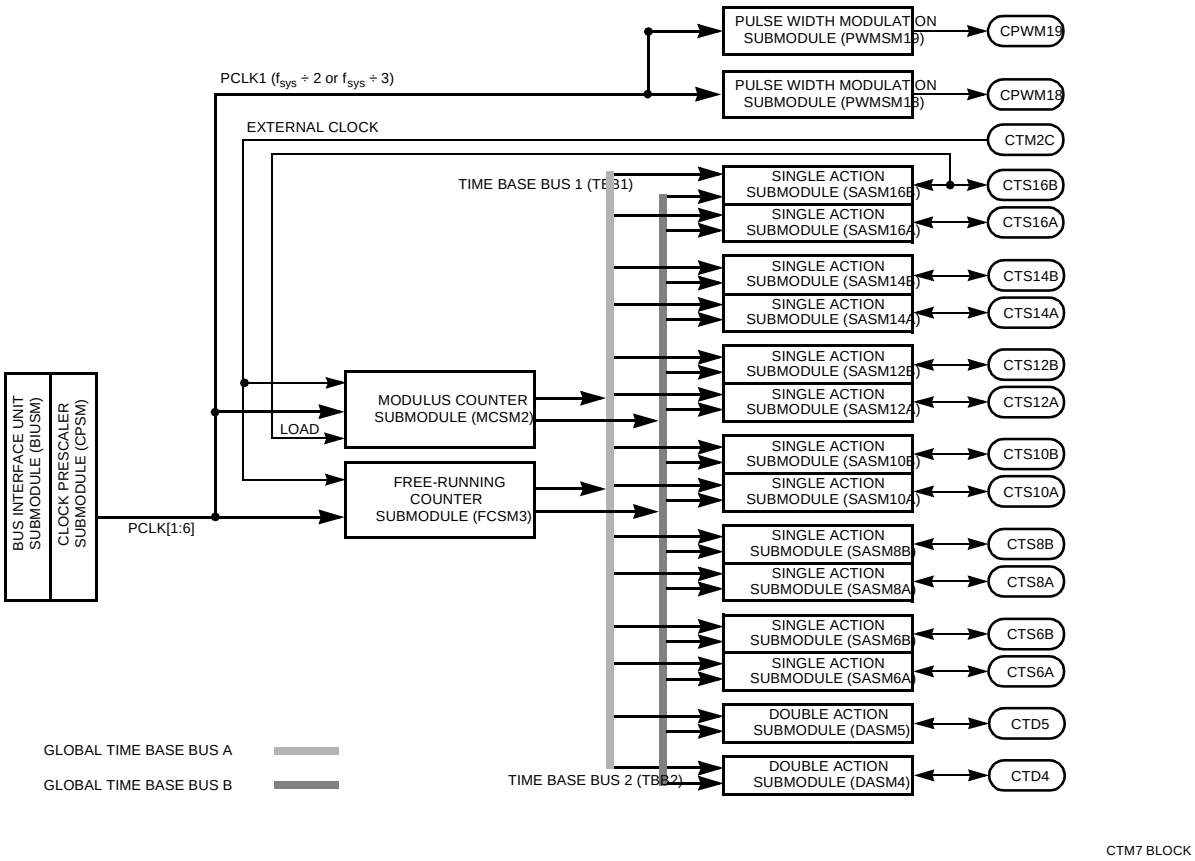


Figure 12-1 CTM7 Block Diagram

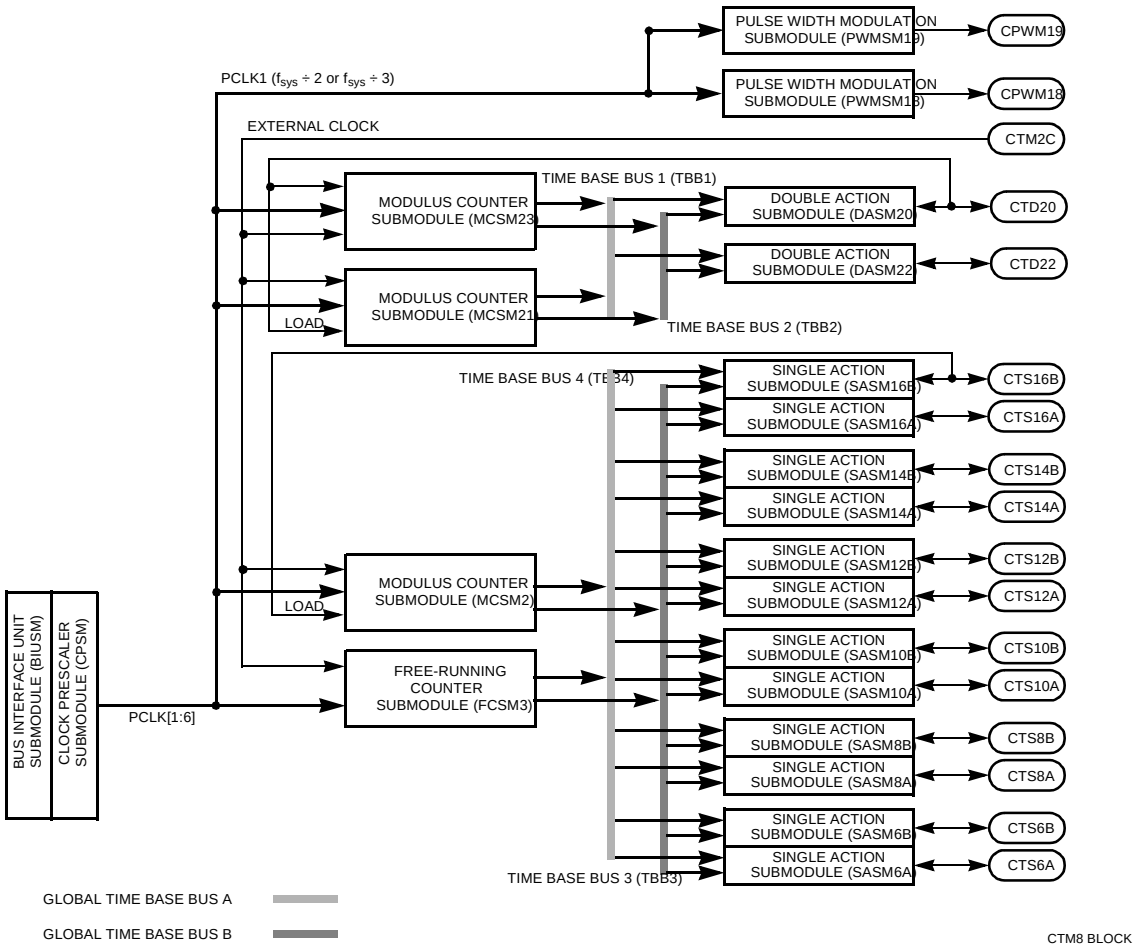


Figure 12-2 CTM8 Block Diagram

The time base buses originate in a counter submodule and are used by the action submodules. Two time base buses are accessible to each submodule.

The bus interface unit submodule (BIUSM) allows all the CTM submodules to pass data to and from the IMB via the submodule bus (SMB).

The counter prescaler submodule (CPSM) generates six different clock frequencies which can be used by any counter submodule. This submodule is contained within the BIUSM.

The free-running counter submodule (FCSM) has a 16-bit up counter with an associated clock source selector, selectable time-base bus drivers, writable control registers, readable status bits, and interrupt logic. The CTM7 and CTM8 have one FCSM.

The modulus counter submodule (MCSM) is an enhanced FCSM. A modulus register gives the additional flexibility of recycling the counter at a count other than 64K clock cycles. The CTM7 has one MCSM; the CTM8 has three MCSMs.

The single action submodule (SASM) provides an input capture and an output compare for each of two bidirectional pins. A total of six SASMs (eight channels) are contained in the CTM7 and the CTM8.

The double-action submodule (DASM) provides two 16-bit input capture or two 16-bit output compare functions that can occur automatically without software intervention. The CTM7 and the CTM8 have two DASMs.

The pulse width modulation submodule (PWMSM) can generate pulse-width modulated signals over a wide range of frequencies, independently of other CTM output signals. PWMSMs are not affected by time base bus activity. The CTM7 and the CTM8 have two PWMSMs.

12.2 Address Map

The CTM address map occupies 256 bytes from address $YFF900 to $YFF9FF. All CTM registers are accessible only when the CPU16 is in supervisor mode. All reserved addresses return zero when read, and writes have no effect. Refer to **APPENDIX D REGISTER SUMMARY** for information concerning CTM7 and CTM8 address map and register bit/field descriptions.

12.3 Time Base Bus System

The CTM7 time base bus system is composed of two 16-bit buses: TBB1 and TBB2. The CTM8 time base bus system is composed of four 16-bit buses: TBB1, TBB2, TBB3, and TBB4. These buses are used to transfer timing information from the counter submodules to the action submodules. Two time base buses are available to each submodule. A counter submodule can drive one of the two time base buses to which it is connected. Each action submodule can choose one of the two time base buses to which it is connected as its time base. Control bits within each CTM submodule select connection to the appropriate time base bus.

The time base buses are precharge/discharge type buses with wired-OR capability. Therefore, no hardware damage occurs when more than one counter drives the same bus at the same time.

In the CTM7, TBB1 and TBB2 are global and accessible to every submodule. In the CTM8, TBB1 and TBB2 form one local time base bus, TBB3 and TBB4 form another local time base bus. **Table 12-1** shows which time base buses are available to each CTM submodule.

Table 12-1 CTM Time Base Bus Allocation

Submodules		Global/Local Time Base Bus Allocation	
CTM7	**CTM8**	**Global Bus A**	**Global Bus B**
MCSM2	—	TBB4	TBB3
—	MCSM21	TBB1	TBB2
—	MCSM23	TBB1	TBB2
FCSM3	FCSM3	TBB4	TBB3
DASM4	—	TBB4	TBB3
DASM5	—	TBB4	TBB3
—	DASM20	TBB1	TBB2
—	DASM22	TBB1	TBB2
SASM6	SASM6	TBB4	TBB3
SASM8	SASM8	TBB4	TBB3
SASM10	SASM10	TBB4	TBB3
SASM12	SASM12	TBB4	TBB3
SASM14	SASM14	TBB4	TBB3
SASM16	SASM16	TBB4	TBB3

Each PWMSM has an independent 16-bit counter and 8-bit prescaler clocked by the PCLK1 signal, which is generated by the CPSM. The PWMSMs are not connected to any of the time base buses. Refer to **12.10 Pulse-Width Modulation Submodule (PWMSM)** for more information.

12.4 Bus Interface Unit Submodule (BIUSM)

The BIUSM connects the SMB to the IMB and allows the CTM submodules to communicate with the CPU16. The BIUSM also communicates CTM submodule interrupt requests to the IMB, and transfers the interrupt level, arbitration number and vector number to the CPU16 during the interrupt acknowledge cycle.

12.4.1 STOP Effect On the BIUSM

When the CPU16 STOP instruction is executed, only the CPU16 is stopped; the CTM continues to operate as normal.

12.4.2 Freeze Effect On the BIUSM

CTM response to assertion of the IMB FREEZE signal is controlled by the FRZ bit in the BIUSM configuration register (BIUMCR). Since the BIUSM propagates FREEZE to the CTM submodules via the SMB, the setting of FRZ affects all CTM submodules.

If the IMB FREEZE signal is asserted and FRZ = 1, all CTM submodules freeze. The following conditions apply when the CTM is frozen:

- All submodule registers can still be accessed.
- The CPSM, FCSM, MCSM, and PWMSM counters stop counting.
- The IN status bit still reflects the state of the FCSM external clock input pin.
- The IN2 status bit still reflects the state of the MCSM external clock input pin, and the IN1 status bit still reflects the state of the MCSM modulus load input pin.
- DASM capture and compare functions are disabled.

- The DASM IN status bit still reflects the state of its associated pin in the DIS, IPWM, IPM, and IC modes. In the OCB, OCAB, and OPWM modes, IN reflects the state of the DASM output flip flop.
- When configured for OCB, OCAB, or OPWM modes, the state of the DASM output flip-flop will remain unchanged.
- The state of the PWMSM output flip-flop will remain unchanged.

If the IMB FREEZE signal is asserted and FRZ = 0, the freeze condition is ignored, and all CTM submodules will continue to operate normally.

12.4.3 LPSTOP Effect on the BIUSM

When the CPU16 LPSTOP instruction is executed, the system clock is stopped. All dependent modules, including the CTM, are shut down until low-power STOP mode is exited.

12.4.4 BIUSM Registers

The BIUSM contains a module configuration register, a time base register, and a test register. The BIUSM register block occupies the first four register locations in CTM register space. All unused bits and reserved address locations return zero when read. Writes to unused bits and reserved address locations have no effect. Refer to **APPENDIX D REGISTER SUMMARY** for information concerning BIUSM register and bit descriptions.

12.5 Counter Prescaler Submodule (CPSM)

The counter prescaler submodule (CPSM) is a programmable divider system that provides CTM counters with a choice of six clock signals (PCLK[1:6]) derived from the main MCU system clock. Five of these frequencies are derived from a fixed divider chain. The divide ratio of the last clock frequency is software selectable from a choice of four divide ratios.

The CPSM is part of the BIUSM. **Figure 12-3** shows a block diagram of the CPSM.

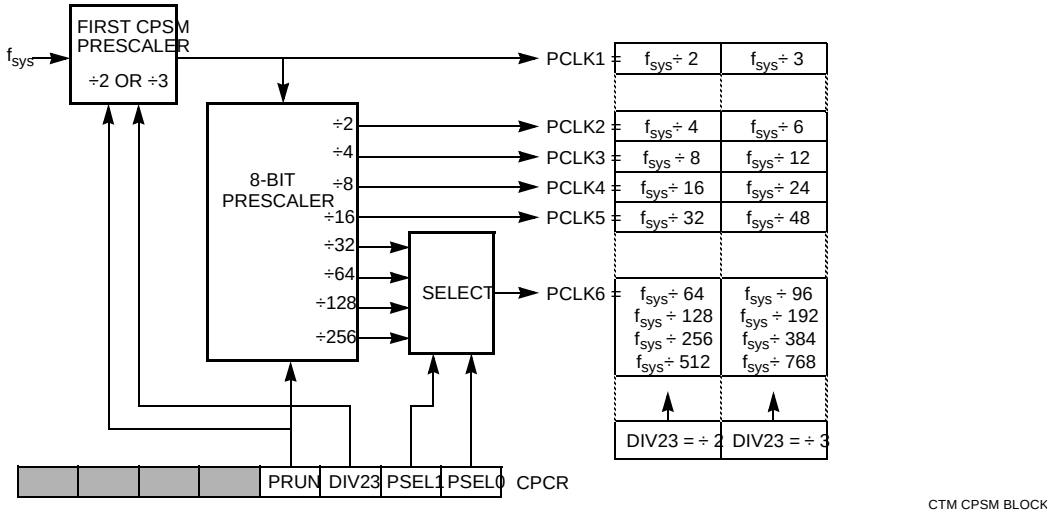


Figure 12-3 CPSM Block Diagram

12.5.1 CPSM Registers

The CPSM contains a control register (CPCR) and a test register (CPTR). All unused bits and reserved address locations return zero when read. Writes to unused bits and reserved address locations have no effect. Refer to **APPENDIX D REGISTER SUMMARY** for information concerning CPSM register and bit descriptions.

12.6 Free-Running Counter Submodule (FCSM)

The free-running counter submodule (FCSM) has a 16-bit up counter with an associated clock source selector, selectable time-base bus drivers, control registers, status bits, and interrupt logic. When the 16-bit up counter overflows from $FFFF to $0000, an optional overflow interrupt request can be generated. The current state of the 16-bit counter is the primary output of the counter submodules. The user can select which, if any, time base bus is to be driven by the 16-bit counter. A software control register selects whether the clock input to the counter is one of the taps from the prescaler or an input pin. The polarity of the external input pin is also programmable.

In order to count, the FCSM requires the CPSM clock signals to be present. After reset, the FCSM does not count until the prescaler in the CPSM starts running (when the software sets the PRUN bit). This allows all counters in the CTM7 submodules to be synchronized.

The CTM7 and the CTM8 have one FCSM. **Figure 12-4** shows a block diagram of the FCSM.

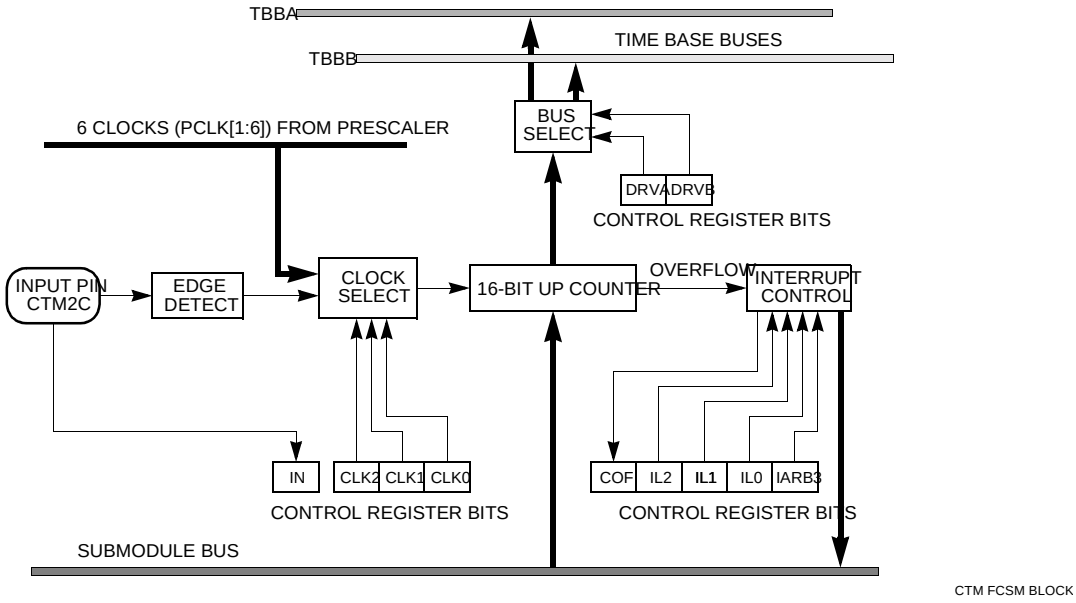


Figure 12-4 FCSM Block Diagram

12.6.1 FCSM Counter

The FCSM counter consists of a 16-bit register and a 16-bit up-counter. Reading the register transfers the contents of the counter to the data bus, while a write to the register loads the counter with a new value. Overflow of the counter is defined to be the transition from $FFFF to $0000. An overflow condition sets the counter overflow flag (COF) in the FCSM status/interrupt/control register (FCSMSIC).

NOTE

Reset presets the counter register to $0000. Writing $0000 to the counter register while the counter's value is $FFFF does not set the COF flag and does not generate an interrupt request.

12.6.2 FCSM Clock Sources

The FCSM has eight software selectable counter clock sources, including:

- Six CPSM prescaler outputs (PCLK[1:6])
- Rising edge on CTM2C input
- Falling edge on the CTM2C input

The clock source is selected by the CLK[2:0] bits in FCSMSIC. When the CLK[2:0] bits are being changed, internal circuitry guarantees that spurious edges occurring on the CTM2C pin do not affect the FCSM. The read-only IN bit in FCSMSIC reflects the state of CTM2C. This pin is Schmitt-triggered and is synchronized with the system clock. The maximum allowable frequency for a clock input on CTM2C is $f_{sys}/4$.

12.6.3 FCSM External Event Counting

When an external clock source is selected, the FCSM can act as an event counter simply by counting the number of events occurring on the CTM2C input pin. Alternatively, the FCSM can be programmed to generate an interrupt request when a predefined number of events have been counted. This is done by presetting the counter with the two's complement value of the desired number of events.

12.6.4 FCSM Time Base Bus Driver

The DRVA and DRVB bits in FCSMSIC select the time base bus to be driven. Which of the time base buses is driven depends on where the FCSM is physically placed in any particular CTM implementation. Refer to **Figure 12-1** and **Table 12-1** for more information.

WARNING

Two time base buses should not be driven at the same time.

12.6.5 FCSM Interrupts

The FCSM can optionally request an interrupt when its counter overflows and the COF bit in FCSMSIC is set. To enable interrupts, set the IL[2:0] field in the FCSMSIC to a non-zero value. The CTM compares the CPU16 IP mask value to the priority of the requested interrupt designated by IL[2:0] to determine whether it should contend for arbitration priority. During arbitration, the BIUSM provides the arbitration value specified by IARB[2:0] in BIUMCR and IARB3 in FCSMSIC. If the CTM wins arbitration, it responds with a vector number generated by concatenating VECT[7:6] in BIUMCR and the six low-order bits specified by the number of the submodule requesting service. Thus, for FCSM3 in the CTM7 , the six low-order bits would be 3 in decimal, or %000011 in binary.

12.6.6 FCSM Registers

The FCSM contains a status/interrupt/control register and a counter register. All unused bits and reserved address locations return zero when read. Writes to unused bits and reserved address locations have no effect. Refer to **APPENDIX D REGISTER SUMMARY** for information concerning FCSM register and bit descriptions.

12.7 Modulus Counter Submodule (MCSM)

The modulus counter submodule (MCSM) is an enhanced FCSM. The MCSM contains a 16-bit modulus latch, a 16-bit loadable up-counter, counter loading logic, a clock selector, selectable time base bus drivers, and an interrupt interface. A modulus register provides the added flexibility of recycling the counter at a count other than 64K clock cycles. The content of the modulus latch is transferred to the counter when an overflow occurs, or when a user-specified edge transition occurs on a designated modulus load input pin. In addition, a write to the modulus counter simultaneously loads both the counter and the modulus latch with the specified value. The counter then begins incrementing from this new value.

In order to count, the MCSM requires the CPSM clock signals to be present. After reset, the MCSM does not count until the prescaler in the CPSM starts running (when the software sets the PRUN bit). This allows all counters in the CTM submodules to be synchronized.

The CTM7 contains one MCSM; the CTM8 contains three MCSMs. **Figure 12-5** shows a block diagram of the MCSM.

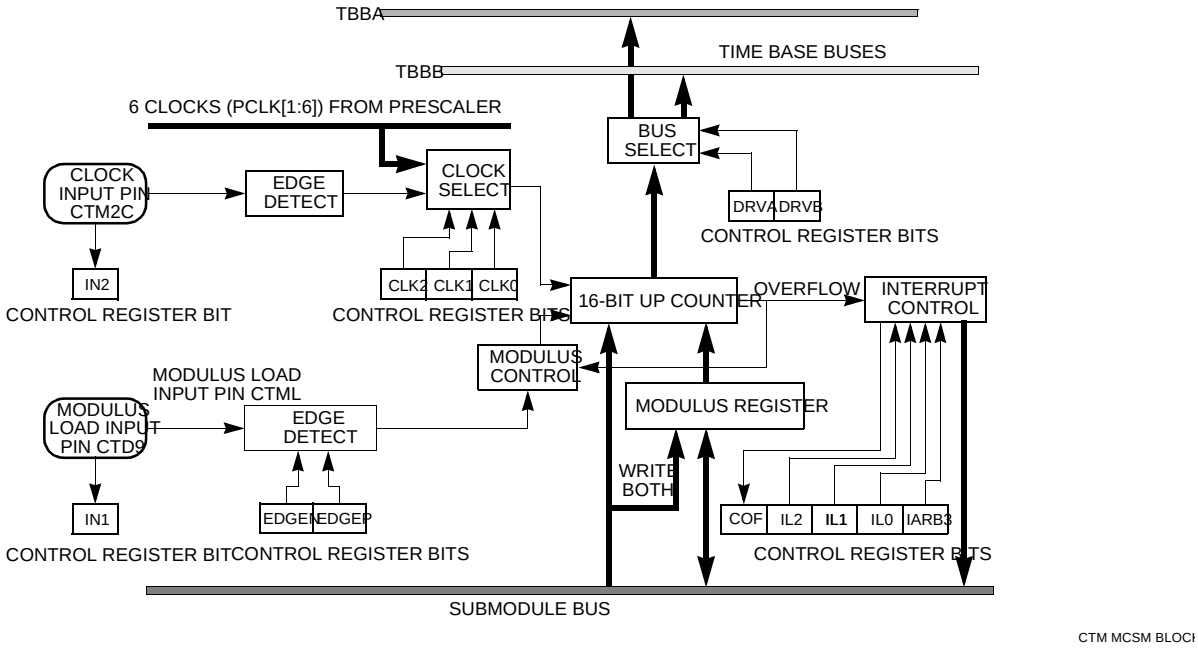


Figure 12-5 MCSM Block Diagram

12.7.1 MCSM Modulus Latch

The 16-bit modulus latch is a read/write register that is used to reload the counter automatically with a predetermined value. The contents of the modulus latch register can be read at any time. Writing to the register loads the modulus latch with the new value. This value is then transferred to the counter register when the next load condition occurs. However, writing to the corresponding counter register loads the modulus latch and the counter register immediately with the new value. The modulus latch register is cleared to $0000 by reset.

12.7.2 MCSM Counter

The counter is composed of a 16-bit read/write register associated with a 16-bit incrementer. Reading the counter transfers the contents of the counter register to the data bus. Writing to the counter loads the modulus latch and the counter register immediately with the new value.

12.7.2.1 Loading the MCSM Counter Register

The MCSM counter is loaded either by writing to the counter register or by loading it from the modulus latch when a counter overflow occurs. Counter overflow will set the COF bit in the MCSM status/interrupt/control register (MCSMSIC).

NOTE

When the modulus latch is loaded with $FFFF, the overflow flag is set on every counter clock pulse.

12.7.2.2 Using the MCSM as a Free-Running Counter

Although the MCSM is a modulus counter, it can operate like a free-running counter by loading the modulus register with $0000.

12.7.3 MCSM Clock Sources

The MCSM has eight software selectable counter clock sources, including:

- Six CPSM prescaler outputs (PCLK[1:6])
- Rising edge on the CTM2C input
- Falling edge on the CTM2C input

The clock source is selected by the CLK[2:0] bits in MCSMSIC. When the CLK[2:0] bits are being changed, internal circuitry guarantees that spurious edges occurring on the CTM2C pin do not affect the MCSM. The read only IN2 bit in MCSMSIC reflects the state of CTM2C. This pin is Schmitt-triggered, and is synchronized with the system clock. The maximum allowable frequency for a clock signal input on CTM2C is $f_{sys}/4$.

12.7.4 MCSM External Event Counting

When an external clock source is selected, the MCSM can act as an event counter simply by counting the number of events occurring on the CTM2C input pin. Alternatively, the MCSM can be programmed to generate an interrupt when a predefined number of events have been counted. This is done by presetting the counter with the two’s complement value of the desired number of events.

12.7.5 MCSM Time Base Bus Driver

The DRVA and DRVB bits in MCSMSIC select the time base bus to be driven. Which of the time base buses is driven depends on where the MCSM is physically placed in any particular CTM implementation. Refer to **Figure 12-1** and **Table 12-1** for more information.

WARNING

Two time base buses should not be driven at the same time.

12.7.6 MCSM Interrupts

The MCSM can optionally request an interrupt when its counter overflows and the COF bit in MCSMSIC is set. To enable interrupts, set the IL[2:0] field in the MCSMSIC to a non-zero value. The CTM compares the CPU16 IP mask value to the priority of the requested interrupt designated by IL[2:0] to determine whether it should contend for arbitration priority. During arbitration, the BIUSM provides the arbitration value specified by IARB[2:0] in BIUMCR and IARB3 in MCSMSIC. If the CTM wins arbitration, it responds with a vector number generated by concatenating VECT[7:6] in BIUMCR and the six low-order bits specified by the number of the submodule requesting service. Thus, for MCSM2 in CTM7, the six low-order bits would be 2 in decimal, or %000010 in binary.

12.7.7 MCSM Registers

The MCSM contains a status/interrupt/control register, a counter, and a modulus latch. All unused bits and reserved address locations return zero when read. Writes to unused bits and reserved address locations have no effect. The CTM7 contains one MCSM, the CTM8 contains three MCSMs. Each MCSM has its own set of registers. Refer to **APPENDIX D REGISTER SUMMARY** for information concerning MCSM register and bit descriptions.

12.8 Single Action Submodule (SASM)

The single action submodule (SASM) provides two identical channels, each having its own input/output pin, but sharing the same interrupt logic, priority level, and arbitration number. Each channel can be configured independently to perform either input capture or output compare. **Table 12-2** shows the different operational modes.

Table 12-2 SASM Operational Modes

Mode	Function
IC[1]	Input capture either on a rising or falling edge or as a read-only input port
OC[2]	Output compare
OCT[2]	Output compare and toggle
OP[2]	Output port

NOTES:
1. When a channel is operating in IC mode, the IN bit in the SIC register reflects the logic state of the corresponding input pin (after being Schmitt triggered and synchronized).
2. When a channel is operating in OC, OCT, or OP mode, the IN bit in the SIC register reflects the logic state of the output of the output flip-flop.

NOTE

All of the functions associated with one pin are called a SASM channel.

The SASM can perform a single timing action (input capture or output compare) before software intervention is required. Each channel includes a 16-bit comparator and one 16-bit register for saving an input capture value or for holding an output compare value. The input edge detector associated with each pin is programmable to cause the capture function to occur on the rising or falling edge. The output flip flop can be set to either toggle when an output compare occurs or to transfer a software provided bit value to the output pin. In either input capture or output compare mode, each channel can be programmed to generate an interrupt. One of the two incoming time-base buses may be selected for each channel. Each channel can also work as a simple I/O pin.

A total of six SASMs (12 channels) are contained in the CTM7 and the CTM8. **Figure 12-6** shows a block diagram of the SASM.

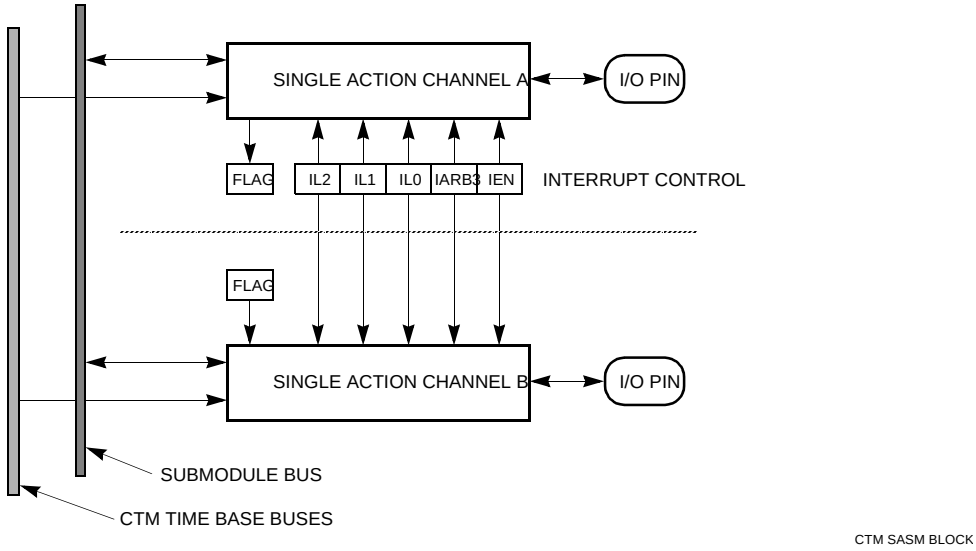


Figure 12-6 SASM Block Diagram

12.8.1 SASM Interrupts

The SASM can optionally request an interrupt when the FLAG bit in SASMSIC is set. To enable interrupts, set the IL[2:0] field in SASMSIC to a non-zero value. The CTM compares the CPU16 IP mask value to the priority of the requested interrupt designated by IL[2:0] to determine whether it should contend for arbitration priority. During arbitration, the BIUSM provides the arbitration value specified by IARB[2:0] in BIUMCR and IARB3 in SASMSIC. If the CTM wins arbitration, it responds with a vector number generated by concatenating VECT[7:6] in BIUMCR and the six low-order bits specified by the number of the submodule requesting service. Thus, for SASM6 in the CTM7, the six low-order bits would be six in decimal, or %000110 in binary.

12.8.2 SASM Registers

The SASM contains one status/interrupt/control register and two data registers (A and B). All unused bits and reserved address locations return zero when read. Writes to unused bits and reserved address locations have no effect. The CTM7 contains six SASMs, each with its own set of registers. Refer to **APPENDIX D REGISTER SUMMARY** for information concerning SASM register and bit descriptions.

12.9 Double-Action Submodule (DASM)

The double-action submodule (DASM) allows two 16-bit input capture or two 16-bit output compare functions to occur automatically without software intervention. The input edge detector can be programmed to trigger the capture function on user-specified edges. The output flip flop can be set by one of the output compare functions, and reset by the other one. Interrupt requests can optionally be generated by the input capture and the output compare functions. The user can select one of two incoming time bases for the input capture and output compare functions.

Six operating modes allow the DASM input capture and output compare functions to perform pulse width measurement, period measurement, single pulse generation, and continuous pulse width modulation, as well as standard input capture and output compare. The DASM can also function as a single I/O pin.

DASM operating mode is determined by the mode select field (MODE[3:0]) in the DASM status/interrupt/control register (DASMSIC). **Table 12-3** shows the different DASM operating modes.

Table 12-3 DASM Modes of Operation

MODE[3:0]	Mode	Description of Mode
0000	DIS	Disabled — Input pin is high impedance; IN gives state of input pin
0001	IPWM	Input pulse width measurement — Capture on leading edge and the trailing edge of an input pulse
0010	IPM	Input period measurement — Capture two consecutive rising/falling edges
0011	IC	Input capture — Capture when the designated edge is detected
0100	OCB	Output compare, flag set on B compare — Generate leading and trailing edges of an output pulse and set the flag
0101	OCAB	Output compare, flag set on A and B compare — Generate leading and trailing edges of an output pulse and set the flag
0110	—	Reserved
0111	—	Reserved
1xxx	OPWM	Output pulse width modulation — Generate continuous PWM output with 7, 9, 11, 12, 13, 14, 15, or 16 bits of resolution

The DASM is composed of two timing channels (A and B), an output flip-flop, an input edge detector, some control logic and an interrupt interface. All control and status bits are contained in DASMSIC.

Channel A consists of one 16-bit data register and one 16-bit comparator. To the user, channel B also appears to consist of one 16-bit data register and one 16-bit comparator, though internally, channel B has two data registers (B1 and B2). DASM operating mode determines which register is software accessible. Refer to **Table 12-4**.

Table 12-4 Channel B Data Register Access

Mode	Data Register
Input Capture (IPWM, IPM, IC)	Registers A and B2 are used to hold the captured values. In these modes, the B1 register is used as a temporary latch for channel B.
Output Compare (OCA, OCAB)	Registers A and B2 are used to define the output pulse. Register B1 is not used in these modes.
Output Pulse Width Modulation Mode (OPWM)	Registers A and B1 are used as primary registers and hidden register B2 is used as a double buffer for channel B.

Register contents are always transferred automatically at the correct time so that the minimum pulse (measured or generated) is just one time base bus count. The A and B data registers are always read/write registers, accessible via the CTM submodule bus.

The CTM7 and the CTM8 have two DASMs. **Figure 12-7** shows a block diagram of the DASM.

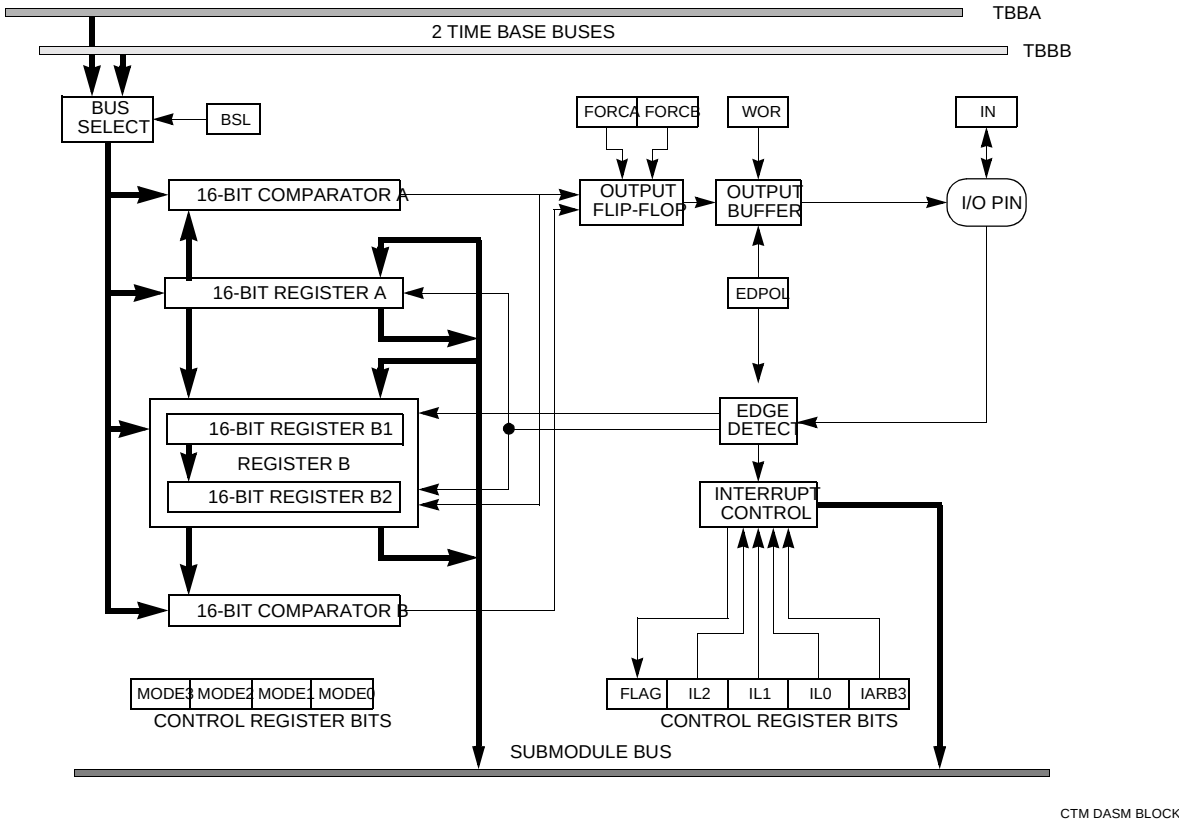


Figure 12-7 DASM Block Diagram

12.9.1 DASM Interrupts

The DASM can optionally request an interrupt when the FLAG bit in DASMSIC is set. To enable interrupts, set the IL[2:0] field in DASMSIC to a non-zero value. The CTM compares the CPU16 IP mask value to the priority of the requested interrupt designated by IL[2:0] to determine whether it should contend for arbitration priority. During arbitration, the BIUSM provides the arbitration value specified by IARB[2:0] in BIUMCR and IARB3 in DASMSIC. If the CTM wins arbitration, it responds with a vector number generated by concatenating VECT[7:6] in BIUMCR and the six low-order bits specified by the number of the submodule requesting service. Thus, for DASM4 in the CTM7, the six low-order bits would be 4 in decimal, or %000100 in binary.

12.9.2 DASM Registers

The DASM contains one status/interrupt/control register and two data registers (A and B). All unused bits and reserved address locations return zero when read. Writes to unused bits and reserved address locations have no effect. The CTM7 and the CTM8 contain two DASMs, each with its own set of registers. Refer to **APPENDIX D REGISTER SUMMARY** for information concerning DASM register and bit descriptions.

12.10 Pulse-Width Modulation Submodule (PWMSM)

The PWMSM allows pulse width modulated signals to be generated over a wide range of frequencies, independently of other CTM output signals. The output pulse width duty cycle can vary from 0% to 100%, with 16 bits of resolution. The minimum pulse width is twice the MCU system clock period. For example, the minimum pulse width is 119 ns when using a 16.78 MHz clock.

The PWMSM is composed of:

- An output flip-flop with output polarity control
- Clock prescaler and selection logic
- A 16-bit up-counter
- Two registers to hold the current and next pulse width values
- Two registers to hold the current and next pulse period values
- A pulse width comparator
- A system state sequencer
- Logic to create 0% and 100% pulses
- Interrupt logic
- A status, interrupt and control register
- A submodule bus interface

The PWMSM includes its own time base counter and does not use the CTM time base buses; however, it does use the prescaled clock signal PCLK1 generated by the CPSM. Refer to **12.5 Counter Prescaler Submodule (CPSM)** and **Figure 12-1** for more information. **Figure 12-8** shows a block diagram of the PWMSM.

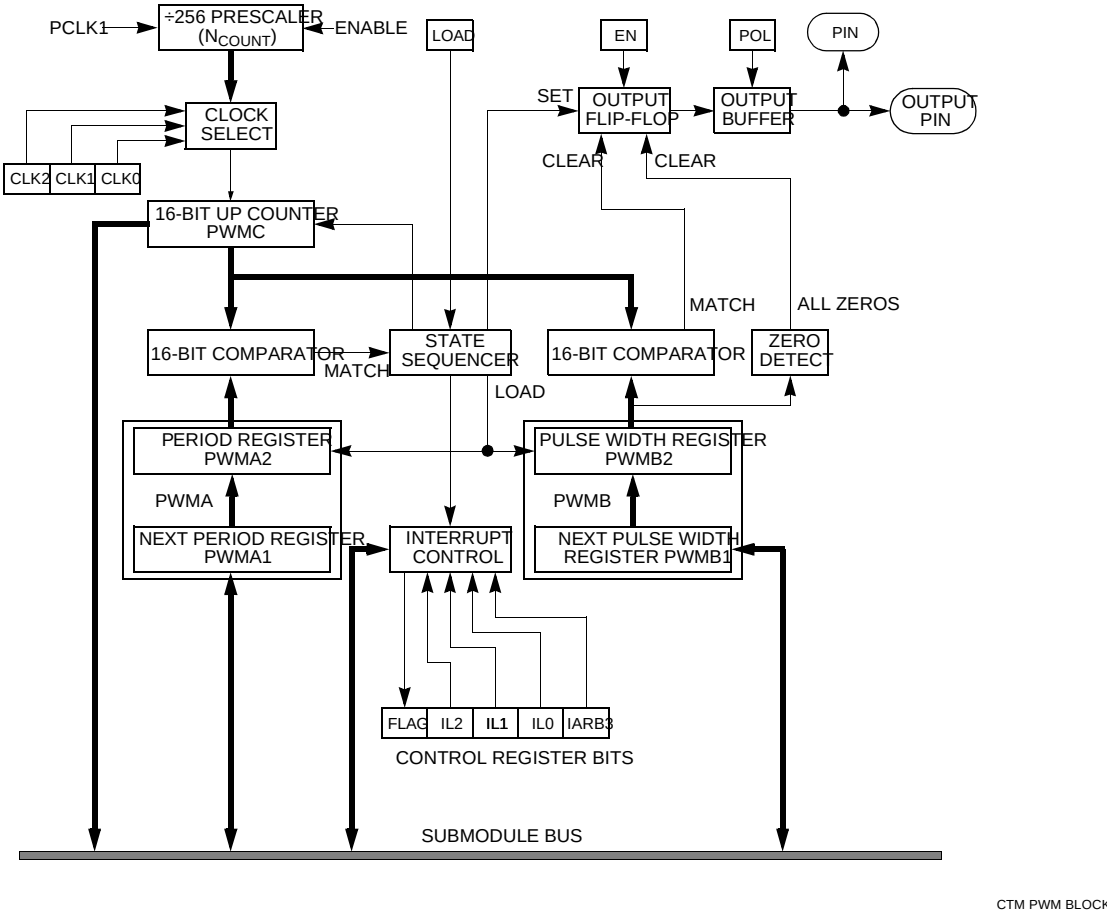


Figure 12-8 Pulse-Width Modulation Submodule Block Diagram

12.10.1 Output Flip-Flop and Pin

The output flip-flop is the basic output mechanism of the PWMSM. Except when the required duty cycle is 0% or 100%, the output flip-flop is set at the beginning of each period and is cleared at the end of the designated pulse width. The polarity of the output pulse is user programmable. The output flip-flop is connected to a buffer that drives the PWMSM's associated output pin. The PWMSM is disabled by clearing the EN bit in the PWMSM status/interrupt/control register (PWMSIC). When the PWMSM is not in use, the output pin can be used as a digital output controlled by the POL bit in PWMSIC.

12.10.2 Clock Selection

The PWMSM contains an 8-bit prescaler that is clocked by the PCLK1 signal ($f_{sys} \div 2$ or $f_{sys} \div 3$) from the CPSM. The CLK[2:0] field in PWMSIC selects which of the eight prescaler outputs drives the PWMSM counter. Refer to **Table 12-5** for the prescaler output.

Table 12-5 PWMSM Divide By Options

CLK2	CLK1	CLK0	PCLK1 = f_{sys} ÷ 2 (CPCR DIV23 = 0)	PCLK1 = f_{sys} ÷ 3 (CPCR DIV23 = 0)
0	0	0	f_{sys} ÷ 2	f_{sys} ÷ 3
0	0	1	f_{sys} ÷ 4	f_{sys} ÷ 6
0	1	0	f_{sys} ÷ 8	f_{sys} ÷ 12
0	1	1	f_{sys} ÷ 16	f_{sys} ÷ 24
1	0	0	f_{sys} ÷ 32	f_{sys} ÷ 48
1	0	1	f_{sys} ÷ 64	f_{sys} ÷ 96
1	1	0	f_{sys} ÷ 128	f_{sys} ÷ 192
1	1	1	f_{sys} ÷ 512	f_{sys} ÷ 768

12.10.3 PWMSM Counter

The 16-bit up counter in the PWMSM provides the time base for the PWM output signal. The counter is held in the $0001 state after reset or when the PWMSM is disabled. When the PWMSM is enabled, the counter begins counting at the rate selected by CLK[2:0] in PWMSIC. Each time the counter matches the contents of the period register, the counter is preset to $0001 and starts to count from that value. The counter can be read at any time from the PWMC register without affecting its value. Writing to the counter has no effect.

12.10.4 PWMSM Period Registers and Comparator

The period section of the PWMSM consists of two 16-bit period registers (PWMA1 and PWMA2) and one 16-bit comparator. PWMA2 holds the current PWM period value, and PWMA1 holds the next PWM period value. The next period of the output PWM signal is established by writing a value into PWMA1. PWMA2 acts as a double buffer for PWMA1, allowing the contents of PWMA1 to be changed at any time without affecting the period of the current output signal. PWMA2 is not user accessible. PWMA1 can be read or written at any time. The new value in PWMA1 is transferred to PWMA2 on the next full cycle of the PWM output or when a one is written to the LOAD bit in PWMSIC.

The comparator continuously compares the contents of PWMA2 with the value in the PWMSM counter. When a match occurs, the state sequencer sets the output flip-flop and resets the counter to $0001.

Period values $0000 and $0001 are special cases. When PWMA2 contains $0000, an output period of 65536 PWM clock periods is generated.

When PWMA2 contains $0001, a period match occurs on every PWM clock period. The counter never increments beyond $0001, and the output level never changes.

NOTE

Values of $0002 in the period register (PWMA2) and $0001 in the pulse width register (PWMB2) result in the maximum possible output frequency for a given PWM counter clock frequency.

12.10.5 PWMSM Pulse-Width Registers and Comparator

The pulse width section of the PWMSM consists of two 16-bit pulse width registers (PWMB1 and PWMB2) and one 16-bit comparator. PWMB2 holds the current PWM pulse width value, and PWMB1 holds the next PWM pulse width value. The next pulse width of the output PWM signal is established by writing a value into PWMB1. PWMB2 acts as a double buffer for PWMB1, allowing the contents of PWMB1 to be changed at any time without affecting the pulse width of the current output signal. PWMB2 is not user accessible. PWMB1 can be read or written at any time. The new value in PWMB1 is transferred to PWMB2 on the next full cycle of the output or when a one is written to the LOAD bit in PWMSIC.

The comparator continuously compares the contents of PWMB2 with the counter. When a match occurs, the output flip-flop is cleared. This pulse width match completes the pulse width, however, it does not affect the counter.

The PWM output pulse may be as short as one PWM counter clock period (PWMB2 = $0001). It may be as long as one PWM clock period less than the PWM period. For example, a pulse width equal to 65535 PWM clock periods can be obtained by setting PWMB2 to $FFFF and PWMA2 to $0000.

12.10.6 PWMSM Coherency

Access to PWMSM registers can be accomplished with 16-bit transfers in most cases. The PWMSM treats a 32-bit access as two 16-bit accesses, except when the access is a write to the period and pulse width registers. A single long word write can set both PWMA1 and PWMB1 because they occupy subsequent memory addresses. If the write can be completed within the current PWM period, there is no visible effect on the output signal. New values loaded into PWMA1 and PWMB1 will be transferred into PWMA2 and PWMB2 at the start of the next period. If the write coincides with the end of the current PWM period, the transfer of values from PWMA1 and PWMB1 into PWMA2 and PWMB2 will be suppressed until the end of the next period. This prevents undesired glitches on the output signal. During the period that is output before the suppressed transfer completes, the current values in PWMA2 and PWMB2 are used.

12.10.7 PWMSM Interrupts

The FLAG bit in PWMSIC is set when a new PWM period begins and indicates that the period and pulse width registers (PWMA1 and PWMB1) may be updated with new values for the next output period. The PWMSM can optionally request an interrupt when FLAG is set. To enable interrupts, set the IL[2:0] field in PWMSIC to a non-zero value. The CTM compares the CPU16 IP mask value to the priority of the requested

interrupt designated by IL[2:0] to determine whether it should contend for arbitration priority.

During arbitration, the BIUSM provides the arbitration value specified by IARB[2:0] in BIUMCR and IARB3 in PWMSIC. If the CTM wins arbitration, it responds with a vector number generated by concatenating VECT[7:6] in BIUMCR and the six low-order bits specified by the number of the submodule requesting service. Thus, for PWMSM18 in the CTM7, the six low-order bits would be 18 in decimal, or %10010 in binary.

12.10.8 PWM Frequency

The relationship between the PWM output frequency (f_{PWM}) and the MCU system clock frequency (f_{sys}) is given by the following equation:

$$f_{PWM} = \frac{f_{sys}}{N_{CLOCK} \cdot N_{PERIOD}}$$

where N_{CLOCK} is the divide ratio specified by the CLK[2:0] field in PWMSIC and N_{PERIOD} is the period specified by PWMA1.

The minimum PWM output frequency achievable with a specified number of bits of resolution for a given system clock frequency is:

$$\text{Minimum } f_{PWM} = \frac{f_{sys}}{256 N_{CPSM} \cdot 2^{\text{Bits of Resolution}}}$$

where N_{CPSM} is the CPSM divide ratio of two or three.

Similarly, the maximum PWM output frequency achievable with a specified number of bits of resolution for a given system clock frequency is:

$$\text{Maximum } f_{PWM} = \frac{f_{sys}}{N_{CPSM} \cdot 2^{\text{Bits of Resolution}}}$$

Tables 12-6 and **12-7** summarize the minimum pulse widths and frequency ranges available from the PWMSM based on the CPSM system clock divide ratio and a system clock frequency of 16.78 MHz.

Table 12-6 PWM Pulse and Frequency Ranges (in Hz) Using ÷ 2 Option (16.78 MHz)

f_{sys} Divide Ratio	Minimum Pulse Width	Bits of Resolution															
		16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
÷ 2	0.119 µs	128	256	512	1024	2048	4096	8192	16384	32768	65.5k	131k	262k	524k	1049k	2097k	4195k
÷ 4	0.238 µs	64	128	256	512	1024	2048	4096	8192	16384	32768	65.5k	131k	262k	524k	1049k	2097k
÷ 8	0.477 µs	32	64	128	256	512	1024	2048	4096	8192	16384	32768	65.5k	131k	262k	524k	1049k
÷ 16	0.954 µs	16	32	64	128	256	512	1024	2048	4096	8192	16384	32768	65.5k	131k	262k	524k
÷ 32	1.91 µs	8.0	16	32	64	128	256	512	1024	2048	4096	8192	16384	32768	65.5k	131k	262k
÷ 64	3.81 µs	4.0	8.0	16	32	64	128	256	512	1024	2048	4096	8192	16384	32768	65.5k	131k
÷ 128	7.63 µs	2.0	4.0	8.0	16	32	64	128	256	512	1024	2048	4096	8192	16384	32768	65.5k
÷ 512	30.5 µs	0.5	1.0	2.0	4.0	8.0	16	32	64	128	256	512	1024	2048	4096	8192	16384

Table 12-7 PWM Pulse and Frequency Ranges (in Hz) Using ÷ 3 Option (16.78 MHz)

f_{sys} Divide Ratio	Minimum Pulse Width	Bits of Resolution															
		16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
÷ 3	0.179 µs	85.33	170.7	341.3	682.7	1365	2731	5461	10923	21845	43.69k	87.38k	174.8k	349.5k	699.1k	1398k	2796k
÷ 6	0.358 µs	42.67	85.33	170.7	341.3	682.7	1365	2731	5461	10923	21845	43.69k	87.38k	174.8k	349.5k	699.1k	1398k
÷ 12	0.715 µs	21.33	42.67	85.33	170.7	341.3	682.7	1365	2731	5461	10923	21845	43.69k	87.38k	174.8k	349.5k	699.1k
÷ 24	1.431 µs	10.67	21.33	42.67	85.33	170.7	341.3	682.7	1365	2731	5461	10923	21845	43.69k	87.38k	174.8k	349.5k
÷ 48	2.861 µs	5.333	10.67	21.33	42.67	85.33	170.7	341.3	682.7	1365	2731	5461	10923	21845	43.69k	87.38k	174.8k
÷ 96	5.722 µs	2.667	5.333	10.67	21.33	42.67	85.33	170.7	341.3	682.7	1365	2731	5461	10923	21845	43.69k	87.38k
÷ 192	11.44 µs	1.333	2.667	5.333	10.67	21.33	42.67	85.33	170.7	341.3	682.7	1365	2731	5461	10923	21845	43.69k
÷ 768	45.78 µs	0.333	0.667	1.333	2.667	5.333	10.67	21.33	42.67	85.33	170.7	341.3	682.7	1365	2731	5461	10923

12.10.9 PWM Pulse Width

The shortest output pulse width (t_{PWMIN}) that can be obtained is given by the following equation:

$$t_{PWMIN} = \frac{N_{CLOCK}}{f_{sys}}$$

The maximum output pulse width (t_{PWMAX}) that can be obtained is given by the following equation:

$$t_{PWMAX} = \frac{N_{CLOCK} \cdot (N_{PERIOD} - 1)}{f_{sys}}$$

12.10.10 PWM Period and Pulse Width Register Values

The value loaded into PWMA1 to obtain a given period is:

$$PWMA1 = \frac{f_{sys}}{N_{CLOCK} \cdot f_{PWM}}$$

The value loaded into PWMB1 to obtain a given duty cycle is:

$$PWMB1 = \frac{1}{t_{PWMIN} \cdot f_{PWM}} = \frac{\text{Duty Cycle \%}}{100} \cdot PWMA1$$

12.10.10.1 PWM Duty Cycle Boundary Cases

PWM duty cycles 0% and 100% are special boundary cases (zero pulse width and infinite pulse width) that are defined by the “always clear” and “always set” states of the output flip-flop.

A zero width pulse is generated by setting PWMB2 to $0000. The output is a true steady state signal. An infinite width pulse is generated by setting PWMB2 equal to or greater than the period value in PWMA2. In both cases, the state of the output pin will remain unchanged at the polarity defined by the POL bit in PWMSIC.

NOTE

A duty cycle of 100% is not possible when the output period is set to 65536 PWM clock periods (which occurs when PWMB2 is set to $0000). In this case, the maximum duty cycle is 99.998% (100 x 65535/65536).

Even when the duty cycle is 0% or 100%, the PWMSM counter continues to count.

12.10.11 PWMSM Registers

The PWMSM contains a status/interrupt/control register, a period register, a pulse width register, and a counter register. All unused bits and reserved address locations return zero when read. Writes to unused bits and reserved address locations have no effect. The CTM7 and the CTM8 contain two PWMSMs, each with its own set of registers. Refer to **APPENDIX D REGISTER SUMMARY** for information concerning PWMSM register and bit descriptions.

12.11 CTM Interrupts

The CTM is able to generate as many as 11 requests for interrupt service. Each submodule capable of requesting an interrupt can do so on any of seven levels. Submodules that can request interrupt service have a 3-bit level number and a 1-bit arbitration number that is user-initialized.

The 3-bit level number selects which of seven interrupt signals on the IMB are driven by that submodule to generate an interrupt request. Of the four priority bits provided by the IMB to the CTM for interrupt arbitration, one of them comes from the chosen submodule, and the BIUSM provides the other three. Thus, the CTM can respond with two of the 15 possible arbitration numbers.

During the IMB arbitration process, the BIUSM manages the separate arbitration among the CTM submodules to determine which submodule should respond. The CTM has a fixed hardware prioritization scheme for all submodules. When two or more submodules have an interrupt request pending at the level being arbitrated on the IMB, the submodule with the lowest number (also the lowest status/interrupt/control register address) is given the highest priority to respond.

If the CTM wins arbitration, it responds with a vector number generated by concatenating VECT[7:6] in BIUMCR and the six low-order bits specified by the number of the submodule requesting service. **Table 12-8** shows the allocation of CTM7 submodule numbers and interrupt vector numbers. **Table 12-9** shows thej allocation of CTM8 submodule numbers and interrupt vector numbers.

Table 12-8 CTM7 Interrupt Priority and Vector/Pin Allocation

Submodule Name	Submodule Number	Submodule Base Address	Submodule Binary Vector Number
BIUSM	0	$YFF900[1]	None
CPSM	1	$YFF908	None
MCSM	2	$YFF910	xx000010[2]
FCSM	3	$YFF918	xx000011
DASM	4	$YFF920	xx000100
DASM	5	$YFF928	xx000101
SASM	6	$YFF930	xx000110
SASM	8	$YFF940	xx001000
SASM	10	$YFF950	xx001010
SASM	12	$YFF960	xx001100
SASM	14	$YFF970	xx001110
SASM	16	$YFF980	xx010000
PWMSM	18	$YFF990	xx010010
PWMSM	19	$YFF998	xx010011

NOTES:
1. Y = M111, where M is the state of the MM bit in SCIMCR (Y = $7 or $F).
2. “xx” represents VECT[7:6] in the BIUSM module configuration register.

Table 12-9 CTM8 Interrupt Priority and Vector/Pin Allocation

Submodule Name	Submodule Number	Submodule Base Address	Submodule Binary Vector Number
BIUSM	0	$YFF900[1]	None
CPSM	1	$YFF908	None
MCSM	2	$YFF910	xx000010[2]
FCSM	3	$YFF918	xx000011
SASM	6	$YFF930	xx000110
SASM	8	$YFF940	xx001000
SASM	10	$YFF950	xx000110
SASM	12	$YFF960	xx001100
SASM	14	$YFF970	xx001110
SASM	16	$YFF980	xx010000
PWSM	18	$YFF990	xx010010
PWSM	19	$YFF998	xx010011
DASM	20	$YFF9A0	xx010100
MCSM	21	$YFF9A8	xx010101
DASM	22	$YFF9B0	xx010110
MCSM	23	$YFF9B8	xx010111

NOTES:

1. Y = M111, where M is the state of the MM bit in SCIMCR (Y = $7 or $F).
2. “xx” represents VECT[7:6] in the BIUSM module configuration register.

APPENDIX A
ELECTRICAL CHARACTERISTICS

Table A-1 Maximum Ratings

Num	Rating	Symbol	Value	Unit
1	Supply Voltage[1, 2, 3]	V_{DD}	– 0.3 to + 6.5	V
2	Input Voltage[1, 2, 3, 4, 5, 7]	V_{in}	– 0.3 to + 6.5	V
3	Instantaneous Maximum Current Single pin limit (applies to all pins)[1, 3, 5, 6]	I_D	25	mA
4	Operating Maximum Current Digital Input Disruptive Current[3, 5, 6, 7, 8] $V_{NEGCLAMP} \approx -0.3$ V $V_{POSCLAMP} \approx V_{DD} + 0.3$ V	I_{ID}	– 500 to + 500	µA
5	Flash EEPROM Program/Erase Supply Voltage[9, 10]	V_{FPE}	(V_{DD} – 0.5) to +12.6	V
6	Operating Temperature Range C Suffix	T_A	T_L to T_H – 40 to + 85	°C
7	Storage Temperature Range	T_{stg}	– 55 to + 150	°C

NOTES:

1. Permanent damage can occur if maximum ratings are exceeded. Exposure to voltages or currents in excess of recommended values affects device reliability. Device modules may not operate normally while being exposed to electrical extremes.
2. Although sections of the device contain circuitry to protect against damage from high static voltages or electrical fields, take normal precautions to avoid exposure to voltages higher than maximum-rated voltages.
3. This parameter is periodically sampled rather than 100% tested.
4. All pins except TSC.
5. Input must be current limited to the value specified. To determine the value of the required current-limiting resistor, calculate resistance values for positive and negative clamp voltages, then use the larger of the two values.
6. Power supply must maintain regulation within operating V_{DD} range during instantaneous and operating maximum current conditions.
7. All functional non-supply pins are internally clamped to V_{SS} for transitions below V_{SS}. All functional pins except EXTAL, TSC, and XFC are internally clamped to V_{DD} for transitions below V_{DD}.
8. Total input current for all digital input-only and all digital input/output pins must not exceed 10 mA. Exceeding this limit can cause disruption of normal operation.
9. V_{FPE} must not be raised to programming level while V_{DD} is below specified minimum value. V_{FPE} must not be reduced below minimum specified value while V_{DD} is applied.
10. Flash EEPROM modules can be damaged by power-on and power-off V_{FPE} transients. Maximum power-on overshoot tolerance is 13.5 V for periods of less than 30 ns.

Table A-2 Typical Ratings

Num	Rating	Symbol	Value	Unit
1	Supply Voltage	V_{DD}	5.0	V
2	Operating Temperature	T_A	25	°C
3	MC68HC16R1 V_{DD} Supply Current RUN LPSTOP, External clock, maximum f_{sys}	I_{DD}	 95 3	 mA mA
3A	MC68HC916R1 V_{DD} Supply Current RUN LPSTOP, External clock, maximum f_{sys}	I_{DD}	 130 5	 mA mA
4	Clock Synthesizer Operating Voltage	V_{DDSYN}	5.0	V
5	V_{DDSYN} Supply Current VCO on, maximum f_{sys} External Clock, maximum f_{sys} LPSTOP, VCO off V_{DD} powered down	I_{DDSYN}	 1.0 4.0 250 50	 mA mA µA µA
6	RAM Standby Voltage	V_{SB}	3.0	V
7	RAM Standby Current Normal RAM operation Standby operation	I_{SB}	 7.0 40	 µA µA
8	MC68HC16R1/R3 Power Dissipation	P_D	500	mW
8A	MC68HC916R1/916R3 Power Dissipation	P_D	675	mW

Table A-3 Thermal Characteristics

Num	Characteristic	Symbol	Value	Unit
1	Thermal Resistance Plastic 132-Pin Surface Mount	Θ_{JA}	38	°C/W

The average chip-junction temperature (TJ) in C can be obtained from:

$$T_J = T_A + (P_D \times \Theta_{JA}) \quad (1)$$

where:

T_A= Ambient Temperature, °C

Θ_{JA}= Package Thermal Resistance, Junction-to-Ambient, °C/W

$P_D = P_{INT} + P_{I/O}$

$P_{INT} = I_{DD} \times V_{DD}$, Watts — Chip Internal Power

$P_{I/O}$= Power Dissipation on Input and Output Pins — User Determined

For most applications $P_{I/O} < P_{INT}$ and can be neglected. An approximate relationship between P_D and T_J (if $P_{I/O}$ is neglected) is:

$$P_D = K \div (T_J + 273°C) \quad (2)$$

Solving equations 1 and 2 for K gives:

$$K = P_D \times (T_A + 273°C) + \Theta_{JA} \times P_D^2 \quad (3)$$

where K is a constant pertaining to the particular part. K can be determined from equation (3) by measuring P_D (at equilibrium) for a known T_A. Using this value of K, the values of P_D and T_J can be obtained by solving equations (1) and (2) iteratively for any value of T_A.

Table A-4 Clock Control Timing

(V_{DD} and V_{DDSYN} = 5.0 Vdc ±10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H)

Num	Characteristic	Symbol	Min	Max	Unit
1	PLL Reference Frequency Range[1]	f_{ref}	3.2	4.2	MHz
2	System Frequency[2] Slow On-Chip PLL System Frequency Fast On-Chip PLL System Frequency External Clock Operation	f_{sys}	dc 4 (f_{ref}) 4 (f_{ref}) /128 dc	16.78 16.78 16.78 16.78	MHz
3	PLL Lock Time[1, 7, 8, 9] Changing W or Y in SYNCR or exiting from LPSTOP[3] Warm Start-up[4] Cold Start-up (fast reference option only)[5]	t_{lpll}	— — —	20 50 75	ms
4	VCO Frequency[6]	f_{VCO}	—	2 (f_{sys} max)	MHz
5	Limp Mode Clock Frequency SYNCR X bit = 0 SYNCR X bit = 1	f_{limp}	— —	f_{sys} max/2 f_{sys} max	MHz
6	CLKOUT Jitter[1, 7, 8, 9, 10] Short term (5 µs interval) Long term (500 µs interval)	J_{clk}	– 0.5 – 0.05	0.5 0.05	%

NOTES:

1. Tested with either a 4.194 MHz reference or a 32.768 kHz reference.
2. All internal registers retain data at 0 Hz.
3. Assumes that V_{DDSYN} and V_{DD} are stable, that an external filter is attached to the XFC pin, and that the crystal oscillator is stable.
4. Assumes that V_{DDSYN} is stable, that an external filter is attached to the XFC pin, and that the crystal oscillator is stable, followed by V_{DD} ramp-up. Lock time is measured from V_{DD} at specified minimum to $\overline{\text{RESET}}$ negated.
5. Cold start is measured from V_{DDSYN} and V_{DD} at specified minimum to $\overline{\text{RESET}}$ negated.
6. Internal VCO frequency (f_{VCO}) is determined by SYNCR W and Y bit values.
   The SYNCR X bit controls a divide-by-two circuit that is not in the synthesizer feedback loop.
   When X = 0, the divider is enabled, and $f_{sys} = f_{VCO} \div 4$.
   When X = 1, the divider is disabled, and $f_{sys} = f_{VCO} \div 2$.
   X must equal one when operating at maximum specified f_{sys}.
7. This parameter is periodically sampled rather than 100% tested.
8. Assumes that a low-leakage external filter network is used to condition clock synthesizer input voltage. Total external resistance from the XFC pin due to external leakage must be greater than 15 MΩ to guarantee this specification. Filter network geometry can vary depending upon operating environment.
9. Proper layout procedures must be followed to achieve specifications.
10. Jitter is the average deviation from the programmed frequency measured over the specified interval at maximum f_{sys}. Measurements are made with the device powered by filtered supplies and clocked by a stable external clock signal. Noise injected into the PLL circuitry via V_{DDSYN} and V_{SS} and variation in crystal oscillator frequency increase the J_{clk} percentage for a given interval. When clock jitter is a critical constraint on control system operation, this parameter should be measured during functional testing of the final system.

Table A-5 DC Characteristics

(V_{DD} and V_{DDSYN} = 5.0 Vdc ±5%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H)

Num	Characteristic	Symbol	Min	Max	Unit
1	Input High Voltage	V_{IH}	0.7 (V_{DD})	V_{DD} + 0.3	V
2	Input Low Voltage	V_{IL}	V_{SS} – 0.3	0.2 (V_{DD})	V
3	Input Hysteresis[1, 2]	V_{HYS}	0.5	—	V
4	Input Leakage Current[3, 4] V_{in} = V_{DD} or V_{SS} All input-only pins except ADC pins	I_{IN}	–2.5	2.5	µA
5	High Impedance (Off-State) Leakage Current[4, 5] V_{in} = V_{DD} or V_{SS} All input/output and output pins	I_{OZ}	–2.5	2.5	µA
6	CMOS Output High Voltage[4, 6, 7] I_{OH} = –10.0 µA Group 1, 2, 4 input/output and all output pins	V_{OH}	V_{DD} –0.2	—	V
7	CMOS Output Low Voltage[4, 8] I_{OL} = 10.0 µA Group 1, 2, 4 input/output and all output pins	V_{OL}	—	0.2	V
8	Output High Voltage[4, 6, 7] I_{OH} = –0.8 mA Group 1, 2, 4 input/output and all output pins	V_{OH}	V_{DD} –0.8	—	V
9	Output Low Voltage[4, 8] I_{OL} = 1.6 mA Group1 I/O Pins, CLKOUT, FREEZE/QUOT, IPIPE0 I_{OL} = 5.3 mA Group 2 and Group 4 I/O Pins, $\overline{BG}$/$\overline{CSM}$ I_{OL} = 12 mA Group 3	V_{OL}	— — —	0.4 0.4 0.4	V
10	Three State Control Input High Voltage	V_{IIHTSC}	1.6 (V_{DD})	9.1	V
11	Data Bus Mode Select Pull-up Current[9, 10] V_{in} = V_{IL} DATA[15:0] V_{in} = V_{IH} DATA[15:0]	I_{MSP}	— –15	–120 —	µA
12	MC68HC16R1/R3 V_{DD} Supply Current[11, 12, 13] Run LPSTOP, crystal, VCO Off (STSCIM = 0) LPSTOP, external clock input frequency = maximum f_{sys}	I_{DD}	— — —	125 TBD 10	mA µA mA
12A	MC68HC916R1/916R3 V_{DD} Supply Current[11, 12, 13] Run LPSTOP, crystal, VCO Off (STSCIM = 0) LPSTOP, external clock input frequency = maximum f_{sys}	I_{DD}	— — —	160 TBD 10	mA µA mA
13	Clock Synthesizer Operating Voltage	V_{DDSYN}	4.5	5.5	V
14	V_{DDSYN} Supply Current[11, 13] VCO on, 4.195 MHZ crystal reference, maximum f_{sys} External Clock, maximum f_{sys} VCO on, 32.786 kHZ crystal reference, maximum f_{sys} External Clock, maximum f_{sys} LPSTOP, 4.195 MHZ crystal reference, VCO off (STSCIM = 0) 4.195 MHZ crystal, V_{DD} powered down LPSTOP, 32.768 kHZ crystal reference, VCO off (STSCIM = 0) 32.768 kHZ crystal, V_{DD} powered down	I_{DDSYN}	— — — — — — — — —	2 7 TBD TBD 2 2 TBD TBD	mA mA mA mA µA µA µA µA
15	RAM Standby Voltage[14] Specified V_{DD} applied V_{DD} = V_{SS}	V_{SB}	0.0 3.0	5.5 5.5	V

Table A-5 DC Characteristics (Continued)

(V_{DD} and V_{DDSYN} = 5.0 Vdc ±5%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H)

Num	Characteristic	Symbol	Min	Max	Unit
16	RAM Standby Current[12] Normal RAM operation[15] $V_{DD} > V_{SB} - 0.5$ V Transient condition $V_{SB} - 0.5$ V ≥ V_{DD} ≥ V_{SS} + 0.5 V Standby operation[14] $V_{DD} < V_{SS} + 0.5$ V	I_{SB}	— — —	50 3 100	µA mA µA
17	MC68HC16R1/R3 Power Dissipation[16]	P_D	—	732	mW
17A	MC68HC916R1/916R3 Power Dissipation[16]	P_D	—	924	mW
18	Input Capacitance[3, 7, 13] All input-only pins except ADC pins All input/output pins	C_{IN}	— —	10 20	pF
19	Load Capacitance[4] Group 1 I/O Pins, CLKOUT, FREEZE/QUOT, IPIPE0 Group 2 I/O Pins and $\overline{\text{CSBOOT}}$, $\overline{\text{BG}}$/$\overline{\text{CS}}$ Group 3 I/O Pins Group 4 I/O Pins	C_L	— — — —	90 100 130 200	pF

NOTES:

1. Applies to :
   Port ADA[7:0] — AN[7:0]
   Port E[7:4] — $\overline{\text{SIZ}}$[1:0], $\overline{\text{AS}}$, $\overline{\text{DS}}$
   Port F[7:0] — $\overline{\text{IRQ}}$[7:1], MODCLK
   Port MCCI[7:0] — TXDA, TXDB, PMC[3:0], $\overline{\text{BKPT}}$/DSCLK, DSI/IPIPE1, $\overline{\text{RESET}}$, RXD, TSC, EXTAL (when PLL enabled)
2. This parameter is periodically sampled rather than 100% tested.
3. Applies to all input-only pins except ADC pins.
4. Input-Only Pins: EXTAL, TSC, $\overline{\text{BKPT}}$/DSCLK, RXDA/RXDB
   Output-Only Pins:
   Group 1: Port C[6:0] — ADDR[22:19]/$\overline{\text{CS}}$[9:6], FC[2:0]/$\overline{\text{CS5}}$, $\overline{\text{CS3}}$
   Port E[7:0] — $\overline{\text{SIZ}}$[1:0], $\overline{\text{AS}}$, $\overline{\text{DS}}$, $\overline{\text{AVEC}}$, $\overline{\text{DSACK}}$[1:0]
   Port F[7:0] — $\overline{\text{IRQ}}$[7:1], MODCLK
   Port PMC[7:3] — TXDA/TXDB, PCS[3:0]
   ADDR23/$\overline{\text{CS10}}$/ECLK, ADDR[18:0], R/$\overline{\text{W}}$, $\overline{\text{BERR}}$, $\overline{\text{BR}}$/$\overline{\text{CS0}}$, $\overline{\text{BGACK}}$/$\overline{\text{CSE}}$
5. Applies to all input/output and output pins.
6. Does not apply to $\overline{\text{HALT}}$ and $\overline{\text{RESET}}$ because they are open drain pins. Does not apply to Port MCCI[7:0] (TXDA/TXDB, PMC[3:0]) in wired-OR mode.
7. Applies to Group 1, 2, 4 input/output and all output pins.
8. Applies to Group 1, 2, 3, 4 input/output pins, $\overline{\text{BG}}$/$\overline{\text{CS}}$, CLKOUT, $\overline{\text{CSBOOT}}$, FREEZE/QUOT, and IPIPE0.
9. Applies to DATA[15:0].
10. Use of an active pulldown device is recommended.
11. Total operating current is the sum of the appropriate I_{DD}, I_{DDSYN}, I_{SB}, and I_{DDA}.
12. Current measured at maximum system clock frequency, all modules active.
13. M68HC16 R-series MCUscan be ordered with either a a 32.768 kHz crystal reference or a 4.194 MHz crystal reference as a mask option.
14. The RAM module will not switch into standby mode as long as V_{SB} does not exceed V_{DD} by more than 0.5 volts. The RAM array cannot be accessed while the module is in standby mode.
15. When V_{SB} is more than 0.3 V greater than V_{DD}, current flows between the V_{STBY} and V_{DD} pins, which causes standby current to increase toward the maximum transient condition specification. System noise on the V_{DD} and V_{STBY} pin can contribute to this condition.
16. Power dissipation measured with system clock frequency of 16.78 MHz, all modules active. Power dissipation can be calculated using the following expression:

$$P_D = \text{Maximum } V_{DD} (I_{DD} + I_{DDSYN} + I_{SB}) + \text{Maximum } V_{DDA} (I_{DDA})$$

I_{DD} includes supply currents for all device modules powered by V_{DD} pins.

Table A-6 AC Timing

(V_{DD} and V_{DDSYN} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H)[1]

Num	Characteristic	Symbol	Min	Max	Unit
F1	Frequency of Operation	f_{sys}	—	16.78	MHz
1	Clock Period	t_{cyc}	59.6	—	ns
1A	ECLK Period	t_{Ecyc}	476	—	ns
1B	External Clock Input Period[2]	t_{Xcyc}	59.6	—	ns
2, 3	Clock Pulse Width[3]	t_{CW}	24	—	ns
2A, 3A	ECLK Pulse Width	t_{ECW}	236	—	ns
2B, 3B	External Clock Input High/Low Time[2]	t_{XCHL}	29.8	—	ns
4, 5	CLKOUT Rise and Fall Time	t_{Crf}	—	5	ns
4A, 5A	Rise and Fall Time (All Outputs except CLKOUT)	t_{rf}	—	8	ns
4B, 5B	External Clock Input Rise and Fall Time[3]	t_{XCrf}	—	5	ns
6	Clock High to ADDR, FC, SIZE Valid[4]	t_{CHAV}	0	29	ns
7	Clock High to ADDR, Data, FC, SIZE, High Impedance	t_{CHAZx}	0	59	ns
8	Clock High to ADDR, FC, SIZE, Invalid	t_{CHAZn}	0	—	ns
9	Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Asserted[4]	t_{CLSA}	2	24	ns
9A	$\overline{AS}$ to $\overline{DS}$ or $\overline{CS}$ Asserted (Read)[5]	t_{STSA}	–15	15	ns
11	ADDR, FC, SIZE Valid to $\overline{AS}$, $\overline{CS}$, (and $\overline{DS}$ Read) Asserted	t_{AVSA}	15	—	ns
12	Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated	t_{CLSN}	2	29	ns
13	$\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated to ADDR, FC SIZE Invalid (Address Hold)	t_{SNAI}	15	—	ns
14	$\overline{AS}$, $\overline{CS}$ (and $\overline{DS}$ Read) Width Asserted	t_{SWA}	100	—	ns
14A	$\overline{DS}$, $\overline{CS}$ Width Asserted (Write)	t_{SWAW}	45	—	ns
14B	$\overline{AS}$, $\overline{CS}$ (and $\overline{DS}$ Read) Width Asserted (Fast Cycle)	t_{SWDW}	40	—	ns
15	$\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Width Negated[6]	t_{SN}	40	—	ns
16	Clock High to $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$ High Impedance	t_{CHSZ}	—	59	ns
17	$\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated to R/$\overline{W}$ High	t_{SNRN}	15	—	ns
18	Clock High to R/$\overline{W}$ High	t_{CHRH}	0	29	ns
20	Clock High to R/$\overline{W}$ Low	t_{CHRL}	0	29	ns
21	R/$\overline{W}$ High to $\overline{AS}$, $\overline{CS}$ Asserted	t_{RAAA}	15	—	ns
22	R/$\overline{W}$ Low to $\overline{DS}$, $\overline{CS}$ Asserted (Write)	t_{RASA}	70	—	ns
23	Clock High to Data Out Valid	t_{CHDO}	—	29	ns
24	Data Out Valid to Negating Edge of $\overline{AS}$, $\overline{CS}$ (Fast Write Cycle)	t_{DVASN}	15	—	ns
25	$\overline{DS}$, $\overline{CS}$ Negated to Data Out Invalid (Data Out Hold)	t_{SNDOI}	15	—	ns
26	Data Out Valid to $\overline{DS}$, $\overline{CS}$ Asserted (Write)	t_{DVSA}	15	—	ns
27	Data In Valid to Clock Low (Data Setup)[4]	t_{DICL}	5	—	ns

Table A-6 AC Timing (Continued)

(V_{DD} and V_{DDSYN} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H)[1]

Num	Characteristic	Symbol	Min	Max	Unit
27A	Late $\overline{BERR}$, $\overline{HALT}$ Asserted to Clock Low (Setup Time)	t_{BELCL}	20	—	ns
28	$\overline{AS}$, $\overline{DS}$ Negated to $\overline{DSACK[1:0]}$, $\overline{BERR}$, $\overline{HALT}$, $\overline{AVEC}$ Negated	t_{SNDN}	0	80	ns
29	$\overline{DS}$, $\overline{CS}$ Negated to Data In Invalid (Data In Hold)[7]	t_{SNDI}	0	—	ns
29A	$\overline{DS}$, $\overline{CS}$ Negated to Data In High Impedance[7, 8]	t_{SHDI}	—	55	ns
30	CLKOUT Low to Data In Invalid (Fast Cycle Hold)[7]	t_{CLDI}	15	—	ns
30A	CLKOUT Low to Data In High Impedance[7]	t_{CLDH}	—	90	ns
31	$\overline{DSACK[1:0]}$ Asserted to Data In Valid[9]	t_{DADI}	—	50	ns
33	Clock Low to $\overline{BG}$ Asserted/Negated	t_{CLBAN}	—	29	ns
35	$\overline{BR}$ Asserted to $\overline{BG}$ Asserted[10]	t_{BRAGA}	1	—	t_{cyc}
37	$\overline{BGACK}$ Asserted to $\overline{BG}$ Negated	t_{GAGN}	1	2	t_{cyc}
39	$\overline{BG}$ Width Negated	t_{GH}	2	—	t_{cyc}
39A	$\overline{BG}$ Width Asserted	t_{GA}	1	—	t_{cyc}
46	R/$\overline{W}$ Width Asserted (Write or Read)	t_{RWA}	150	—	ns
46A	R/$\overline{W}$ Width Asserted (Fast Write or Read Cycle)	t_{RWAS}	90	—	ns
47A	Asynchronous Input Setup Time $\overline{BR}$, $\overline{BGACK}$, $\overline{DSACK[1:0]}$, $\overline{BERR}$, $\overline{AVEC}$, $\overline{HALT}$	t_{AIST}	5	—	ns
47B	Asynchronous Input Hold Time	t_{AIHT}	15	—	ns
48	$\overline{DSACK[1:0]}$ Asserted to $\overline{BERR}$, $\overline{HALT}$ Asserted[11]	t_{DABA}	—	30	ns
53	Data Out Hold from Clock High	t_{DOCH}	0	—	ns
54	Clock High to Data Out High Impedance	t_{CHDH}	—	28	ns
55	R/W Asserted to Data Bus Impedance Change	t_{RADC}	40	—	ns
70	Clock Low to Data Bus Driven (Show Cycle)	t_{SCLDD}	0	19	ns
71	Data Setup Time to Clock Low (Show Cycle)	t_{SCLDS}	15	—	ns
72	Data Hold from Clock Low (Show Cycle)	t_{SCLDH}	10	—	ns
73	$\overline{BKPT}$ Input Setup Time	t_{BKST}	15	—	ns
74	$\overline{BKPT}$ Input Hold Time	t_{BKHT}	10	—	ns
75	Mode Select Setup Time (DATA[15:0], MODCLK, $\overline{BKPT}$)	t_{MSS}	20	—	t_{cyc}
76	Mode Select Hold Time (DATA[15:0], MODCLK, $\overline{BKPT}$)	t_{MSH}	0	—	ns
77	$\overline{RESET}$ Assertion Time[12]	t_{RSTA}	4	—	t_{cyc}
78	$\overline{RESET}$ Rise Time[13]	t_{RSTR}	—	10	t_{cyc}
100	CLKOUT High to Phase 1 Asserted[14]	t_{CHP1A}	3	40	ns
101	CLKOUT High to Phase 2 Asserted[14]	t_{CHP2A}	3	40	ns
102	Phase 1 Valid to $\overline{AS}$ or $\overline{DS}$ Asserted[14]	t_{P1VSA}	10	—	ns
103	Phase 2 Valid to $\overline{AS}$ or $\overline{DS}$ Asserted[14]	t_{P2VSN}	10	—	ns

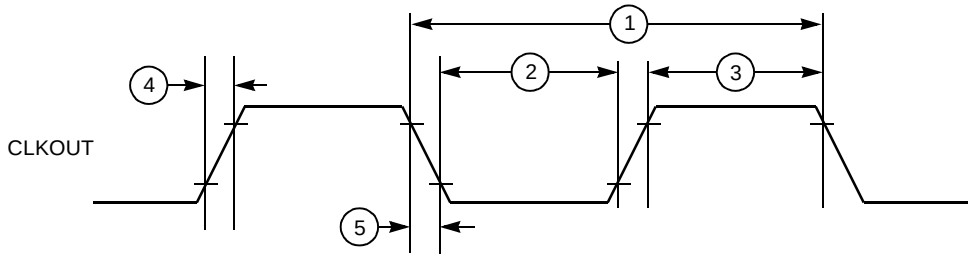
Table A-6 AC Timing (Continued)

(V_{DD} and V_{DDSYN} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H)[1]

Num	Characteristic	Symbol	Min	Max	Unit
104	$\overline{AS}$ or $\overline{DS}$ Valid to Phase 1 Negated[14]	t_{SAP1N}	10	—	ns
105	$\overline{AS}$ or $\overline{DS}$ Negated to Phase 2 Negated[14]	t_{SNP2N}	10	—	ns

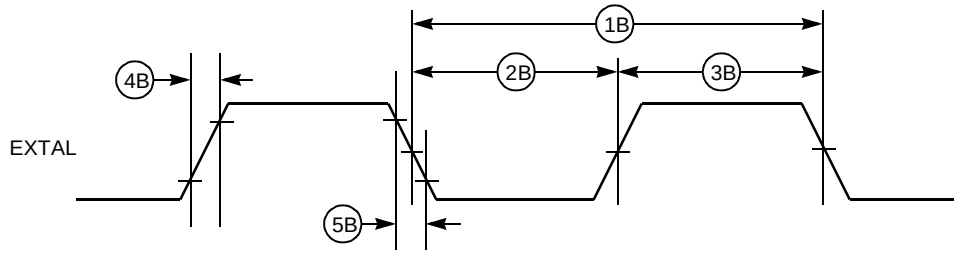
NOTES:

1. All AC timing is shown with respect to V_{IH}/V_{IL} levels unless otherwise noted.
2. When an external clock is used, minimum high and low times are based on a 50% duty cycle. The minimum allowable t_{Xcyc} period is reduced when the duty cycle of the external clock varies. The relationship between external clock input duty cycle and minimum t_{Xcyc} is expressed:
   Minimum t_{Xcyc} period = minimum t_{XCHL} / (50% – external clock input duty cycle tolerance).
3. Parameters for an external clock signal applied while the internal PLL is disabled (MODCLK pin held low during reset) do not pertain to an external reference applied while the PLL is enabled (MODCLK pin held high during reset). When the PLL is enabled, the clock synthesizer detects successive transitions of the reference signal. If transitions occur within the correct clock period, rise/fall times and duty cycle are not critical.
4. Address access time = (2.5 + WS) $t_{cyc} - t_{CHAV} - t_{DICL}$
   Chip select access time = (2 + WS) $t_{cyc} - t_{CLSA} - t_{DICL}$
   Where: WS = number of wait states. When fast termination is used (2 clock bus) WS = –1.
5. Specification 9A is the worst-case skew between $\overline{AS}$ and $\overline{DS}$ or $\overline{CS}$. The amount of skew depends on the relative loading of these signals. When loads are kept within specified limits, skew will not cause $\overline{AS}$ and $\overline{DS}$ to fall outside the limits shown in specification 9.
6. If multiple chip-selects are used, $\overline{CS}$ width negated (specification 15) applies to the time from the negation of a heavily loaded chip select to the assertion of a lightly loaded chip select. The $\overline{CS}$ width negated specification between multiple chip-selects does not apply to chip-selects being used for synchronous ECLK cycles.
7. Hold times are specified with respect to $\overline{DS}$ or $\overline{CS}$ on asynchronous reads and with respect to CLKOUT on fast cycle reads. The user is free to use either hold time.
8. Maximum value is equal to (t_{cyc} / 2) + 25 ns.
9. If the asynchronous setup time (specification 47A) requirements are satisfied, the $\overline{DSACK[1:0]}$ low to data setup time (specification 31) and $\overline{DSACK[1:0]}$ low to $\overline{BERR}$ low setup time (specification 48) can be ignored. The data must only satisfy the data-in to clock low setup time (specification 27) for the following clock cycle. $\overline{BERR}$ must satisfy only the late $\overline{BERR}$ low to clock low setup time (specification 27A) for the following clock cycle.
10. To ensure coherency during every operand transfer, $\overline{BG}$ is not asserted in response to $\overline{BR}$ until after all cycles of the current operand transfer are complete.
11. In the absence of $\overline{DSACK[1:0]}$, $\overline{BERR}$ is an asynchronous input using the asynchronous setup time (specification 47A).
12. After external $\overline{RESET}$ negation is detected, a short transition period (approximately 2 t_{cyc}) elapses, then the SCIM2 drives $\overline{RESET}$ low for 512 t_{cyc}.
13. External logic must pull $\overline{RESET}$ high during this period in order for normal MCU operation to begin.
14. Eight pipeline states are multiplexed into IPIPE[1:0]. The multiplexed signals have two phases.



16 CLKOUT TIM

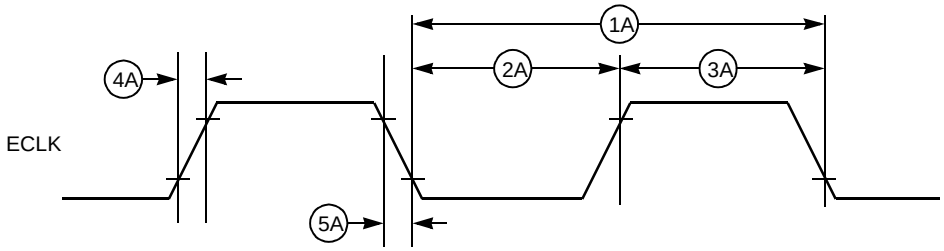
Figure A-1 CLKOUT Output Timing Diagram



NOTE: TIMING SHOWN WITH RESPECT TO V_{IH}/V_{IL} LEVELS.
PULSE WIDTH SHOWN WITH RESPECT TO 50% V_{DD}.

16 EXT CLK INPUT TIM

Figure A-2 External Clock Input Timing Diagram



NOTE: TIMING SHOWN WITH RESPECT TO V_{IH}/V_{IL} LEVELS.

16 ECLK OUTPUT TIM

Figure A-3 ECLK Output Timing Diagram

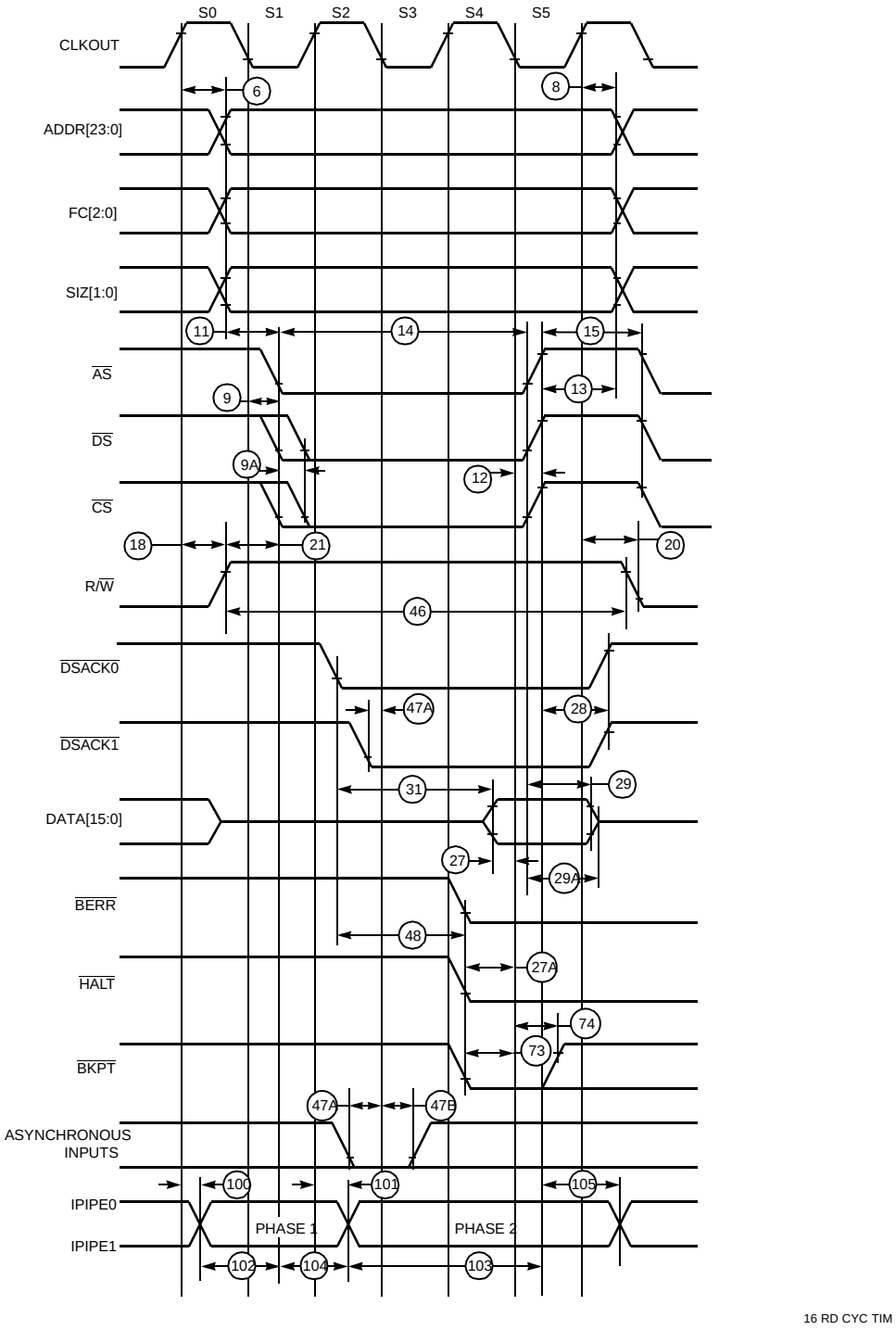


Figure A-4 Read Cycle Timing Diagram

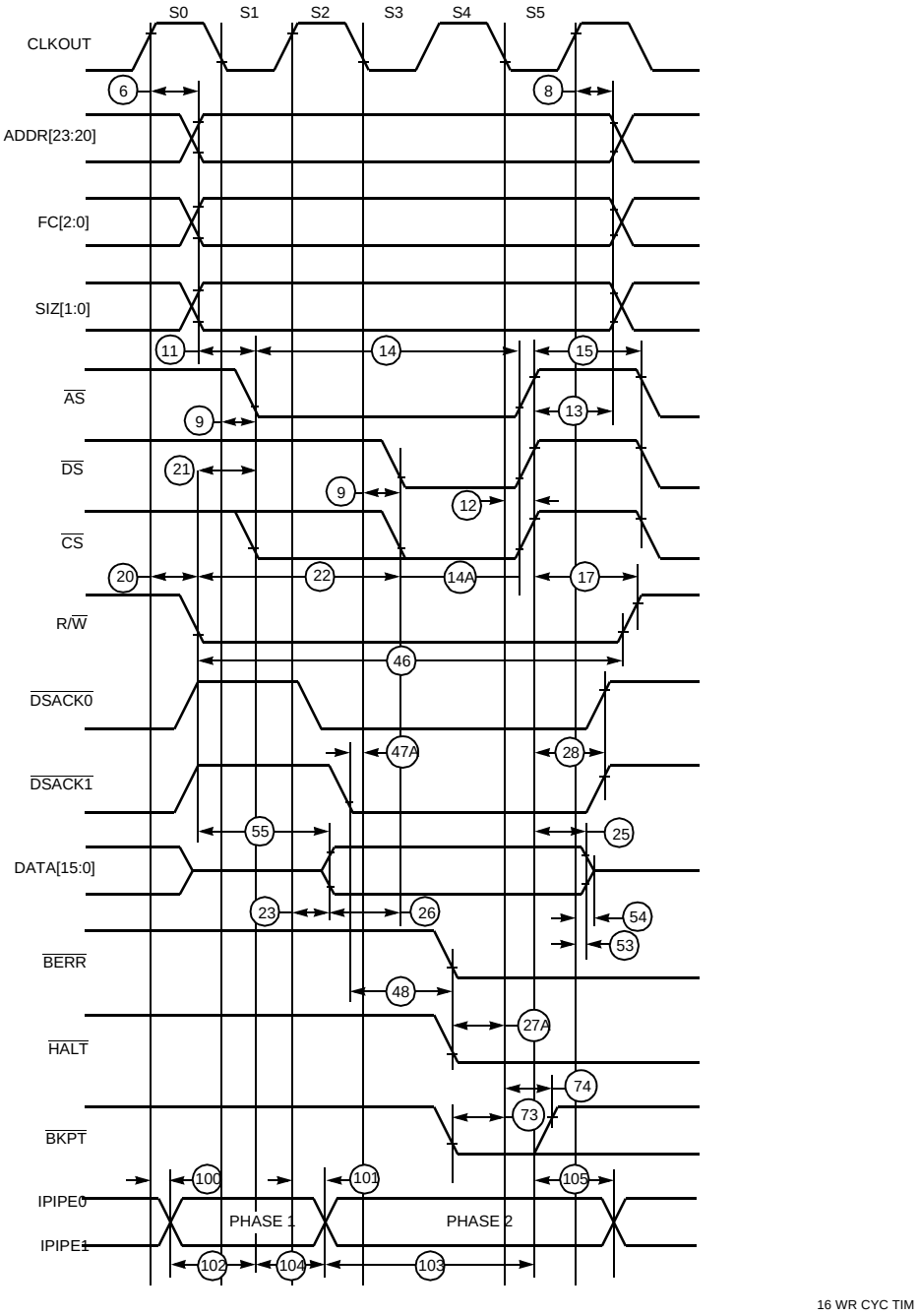


Figure A-5 Write Cycle Timing Diagram

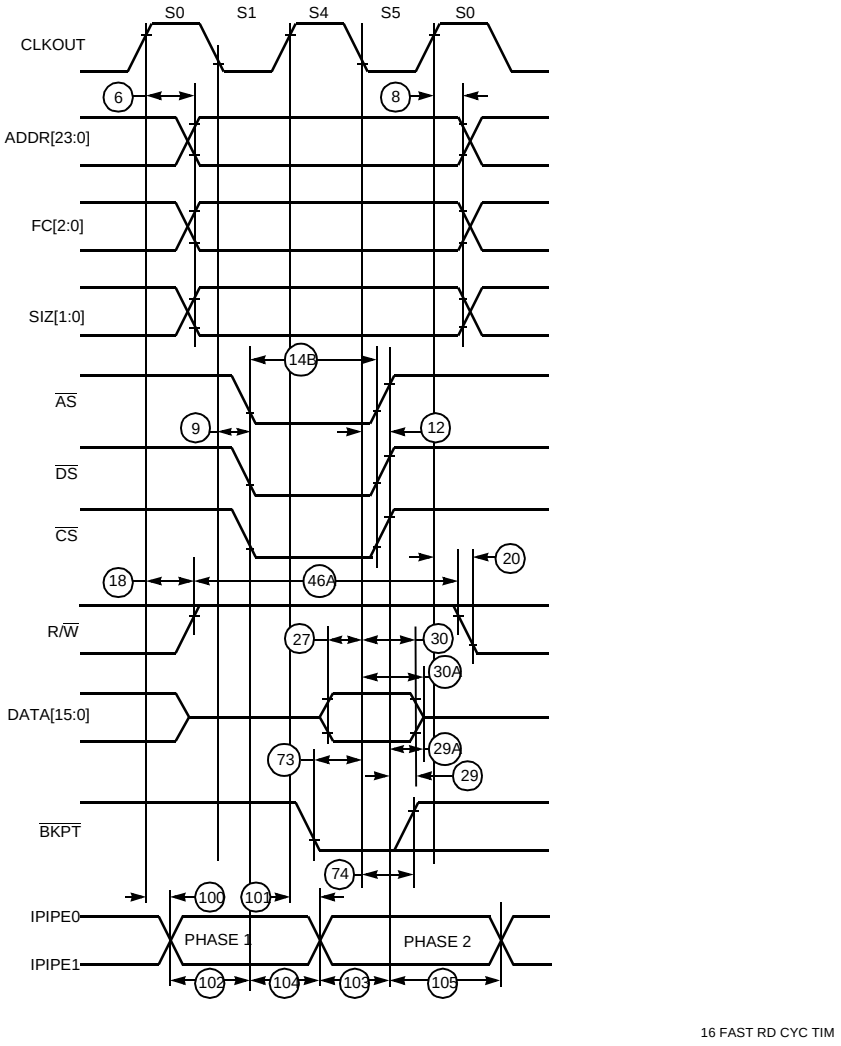


Figure A-6 Fast Termination Read Cycle Timing Diagram

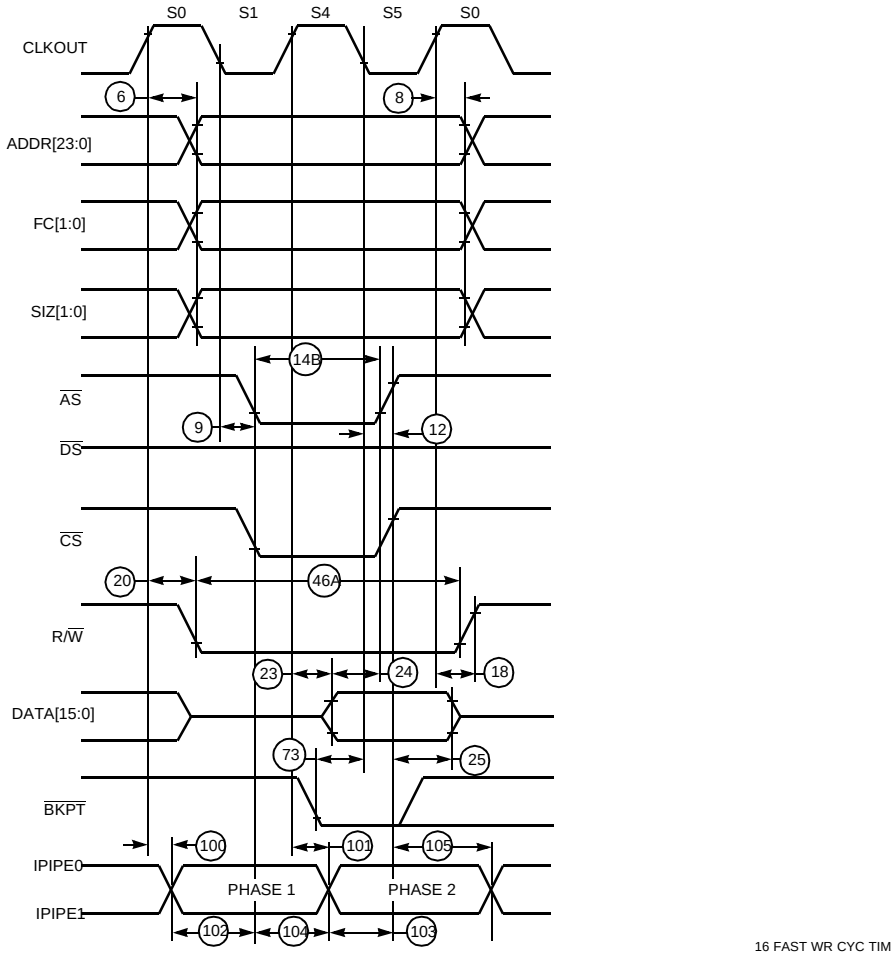


Figure A-7 Fast Termination Write Cycle Timing Diagram

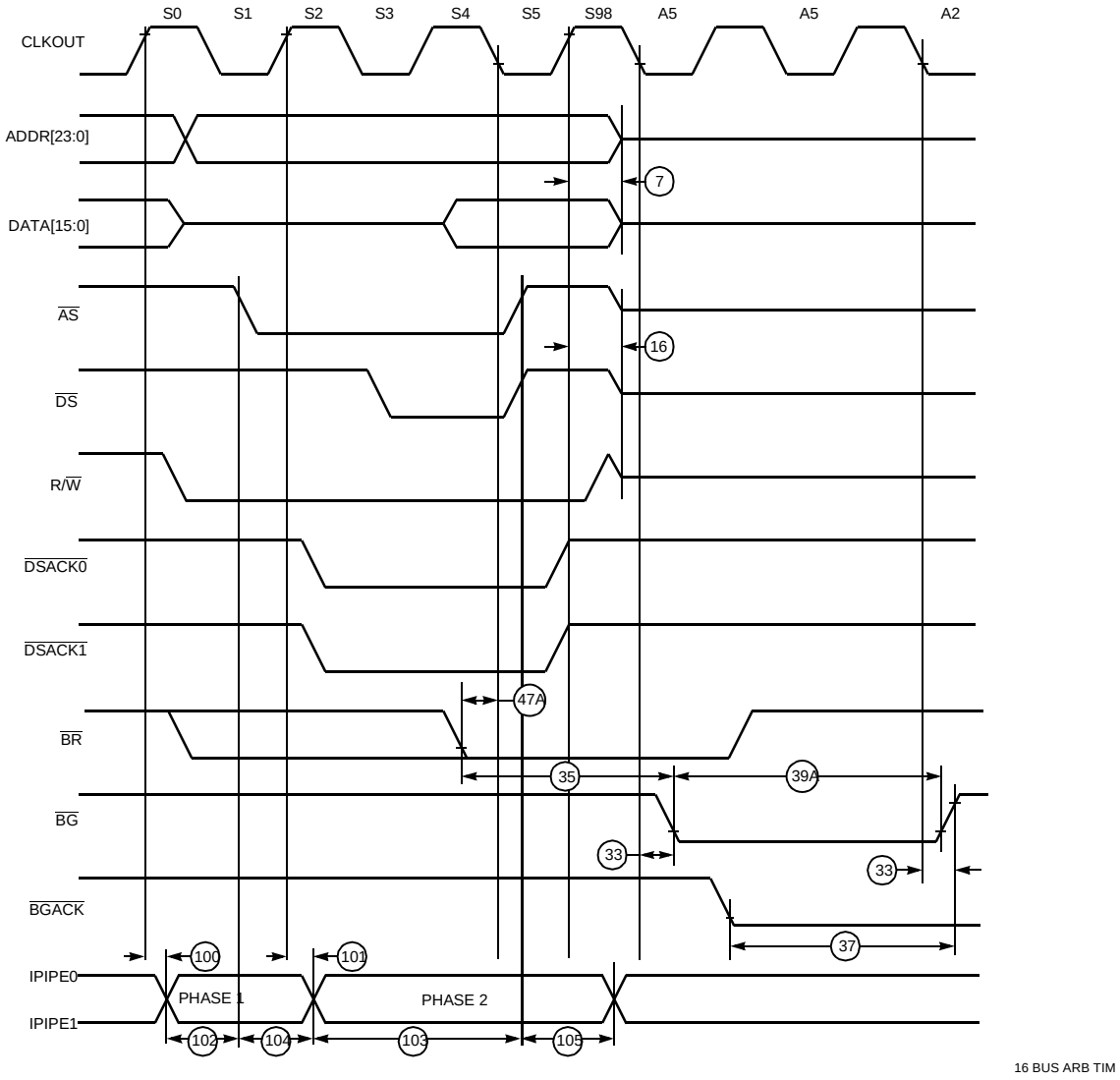


Figure A-8 Bus Arbitration Timing Diagram — Active Bus Case

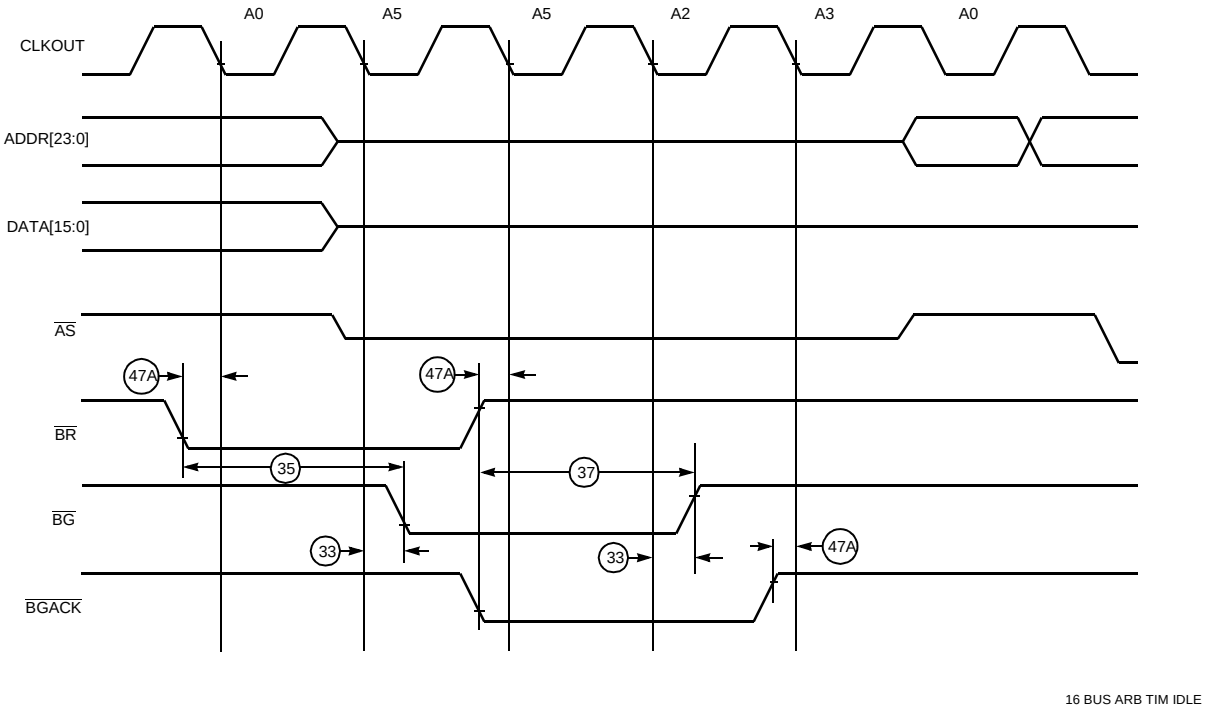
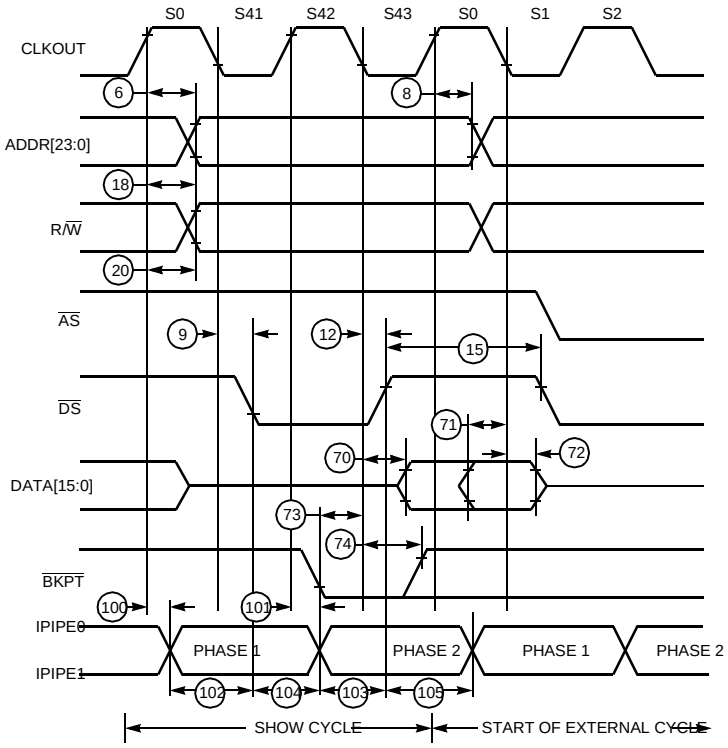


Figure A-9 Bus Arbitration Timing Diagram — Idle Bus Case



NOTE:
SHOW CYCLES CAN STRETCH DURING CLOCK PHASE S42 WHEN BUS ACCESSES TAKE LONGER THAN TWO CYCLES DUE TO IMB MODULE WAIT-STATE INSERTION.

16 SHW CYC TIM

Figure A-10 Show Cycle Timing Diagram

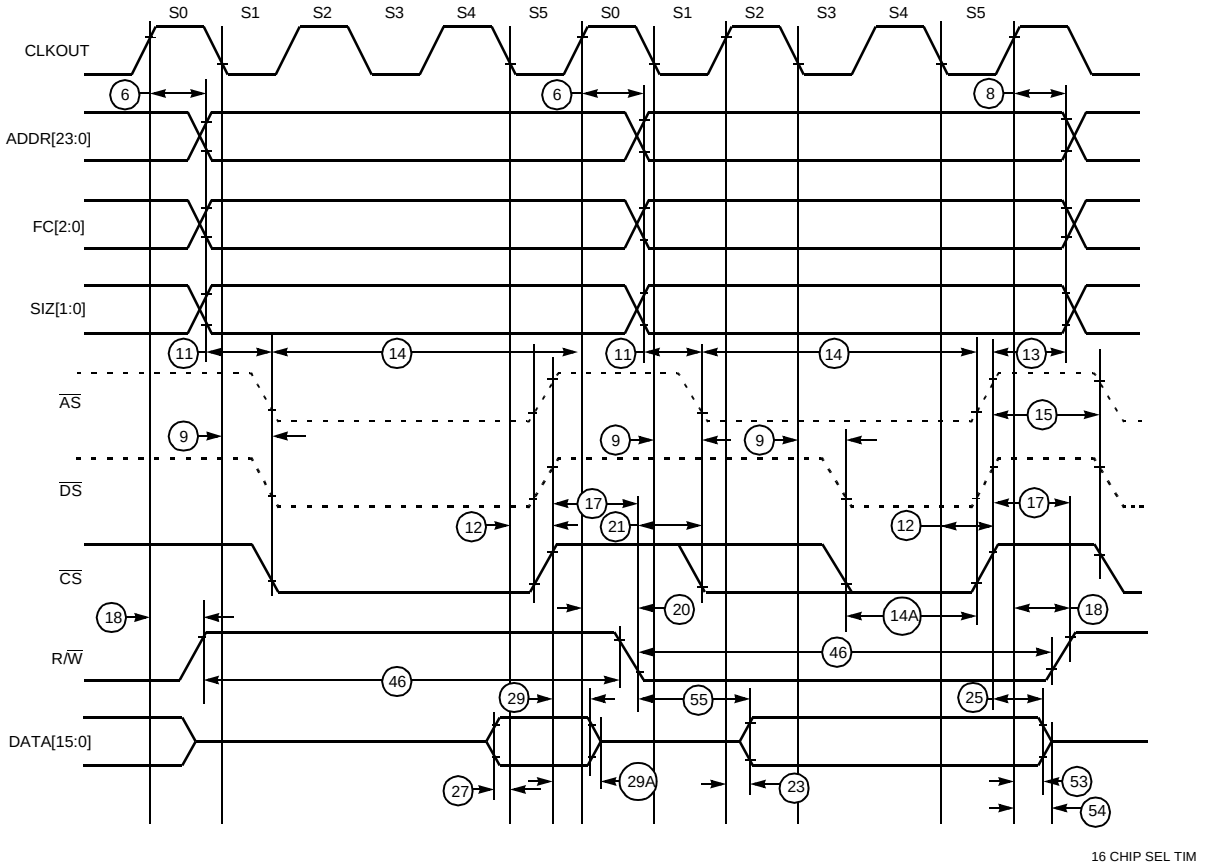


Figure A-11 Chip-Select Timing Diagram

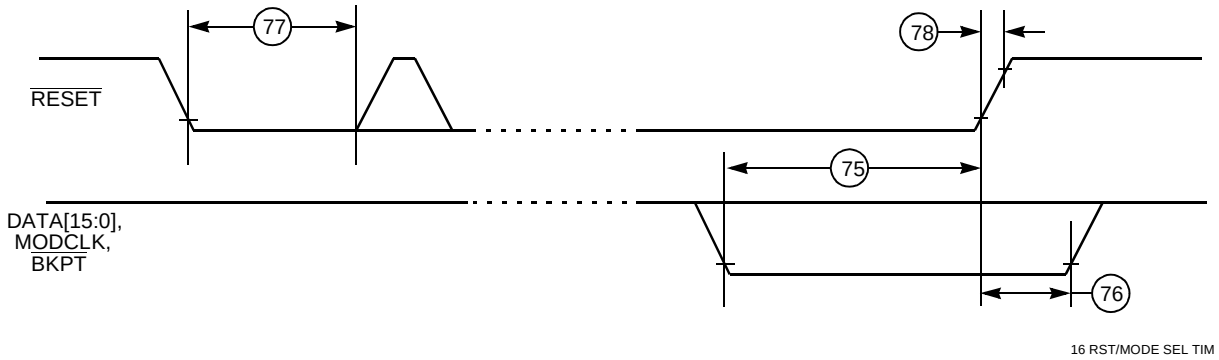


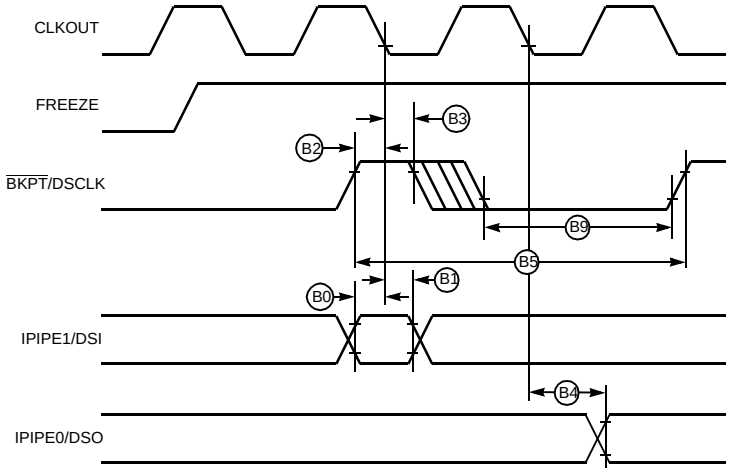
Figure A-12 Reset and Mode Select Timing Diagram

Table A-7 20.97 MHz Background Debug Mode Timing

(V_{DD} and V_{DDSYN} = 5.0 Vdc ± 5%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H)[1]

Num	Characteristic	Symbol	Min	Max	Unit
B0	DSI Input Setup Time	t_{DSISU}	15	—	ns
B1	DSI Input Hold Time	t_{DSIH}	10	—	ns
B2	DSCLK Setup Time	t_{DSCSU}	15	—	ns
B3	DSCLK Hold Time	t_{DSCH}	10	—	ns
B4	DSO Delay Time	t_{DSOD}	—	25	ns
B5	DSCLK Cycle Time	t_{DSCCYC}	2	—	t_{cyc}
B6	CLKOUT High to FREEZE Asserted/Negated	t_{FRZAN}	—	50	ns
B7	CLKOUT High to IPIPE1 High Impedance	t_{IFZ}	—	TBD	ns
B8	CLKOUT High to IPIPE1 Valid	t_{IF}	—	TBD	ns
B9	DSCLK Low Time	t_{DSCLO}	1	—	t_{cyc}
B10	IPIPE1 High Impedance to FREEZE Asserted	t_{IPFA}	TBD	—	t_{cyc}
B11	FREEZE Negated to IPIPE[0:1] Active	t_{FRIP}	TBD	—	t_{cyc}

NOTES:
1. All AC timing is shown with respect to V_{IH}/V_{IL} levels unless otherwise noted.



16 BDM SER COM TIM

Figure A-13 Background Debug Mode Timing Diagram (Serial Communication)

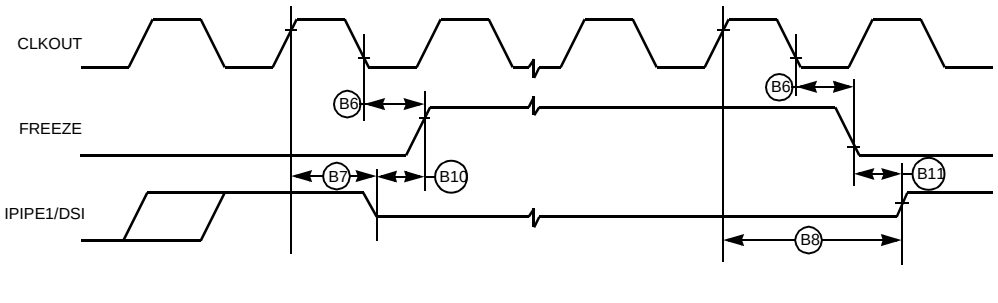


Figure A-14 Background Debug Mode Timing Diagram (Freeze Assertion)

Table A-8 ECLK Bus Timing

(V_{DD} and V_{DDSYN} = 5.0 Vdc ± 5%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H)[1]

Num	Characteristic	Symbol	Min	Max	Unit
E1	ECLK Low to Address Valid[2]	t_{EAD}	—	60	ns
E2	ECLK Low to Address Hold	t_{EAH}	10	—	ns
E3	ECLK Low to $\overline{CS}$ Valid ($\overline{CS}$ Delay)	t_{ECSD}	—	150	ns
E4	ECLK Low to $\overline{CS}$ Hold	t_{ECSH}	15	—	ns
E5	$\overline{CS}$ Negated Width	t_{ECSN}	30	—	ns
E6	Read Data Setup Time	t_{EDSR}	30	—	ns
E7	Read Data Hold Time	t_{EDHR}	15	—	ns
E8	ECLK Low to Data High Impedance	t_{EDHZ}	—	60	ns
E9	$\overline{CS}$ Negated to Data Hold (Read)	t_{ECDH}	0	—	ns
E10	$\overline{CS}$ Negated to Data High Impedance	t_{ECDZ}	—	1	t_{cyc}
E11	ECLK Low to Data Valid (Write)	t_{EDDW}	—	2	t_{cyc}
E12	ECLK Low to Data Hold (Write)	t_{EDHW}	5	—	ns
E13	$\overline{CS}$ Negated to Data Hold (Write)	t_{ECHW}	0	—	ns
E14	Address Access Time (Read)[3]	t_{EACC}	386	—	ns
E15	Chip-Select Access Time (Read)[4]	t_{EACS}	296	—	ns
E16	Address Setup Time	t_{EAS}	—	1/2	t_{cyc}

NOTES:

1. All AC timing is shown with respect to V_{IH}/V_{IL} levels unless otherwise noted.
2. When previous bus cycle is not an ECLK cycle, the address may be valid before ECLK goes low.
3. Address access time = $t_{Ecyc} - t_{EAD} - t_{EDSR}$.
4. Chip select access time = $t_{Ecyc} - t_{ECSD} - t_{EDSR}$.

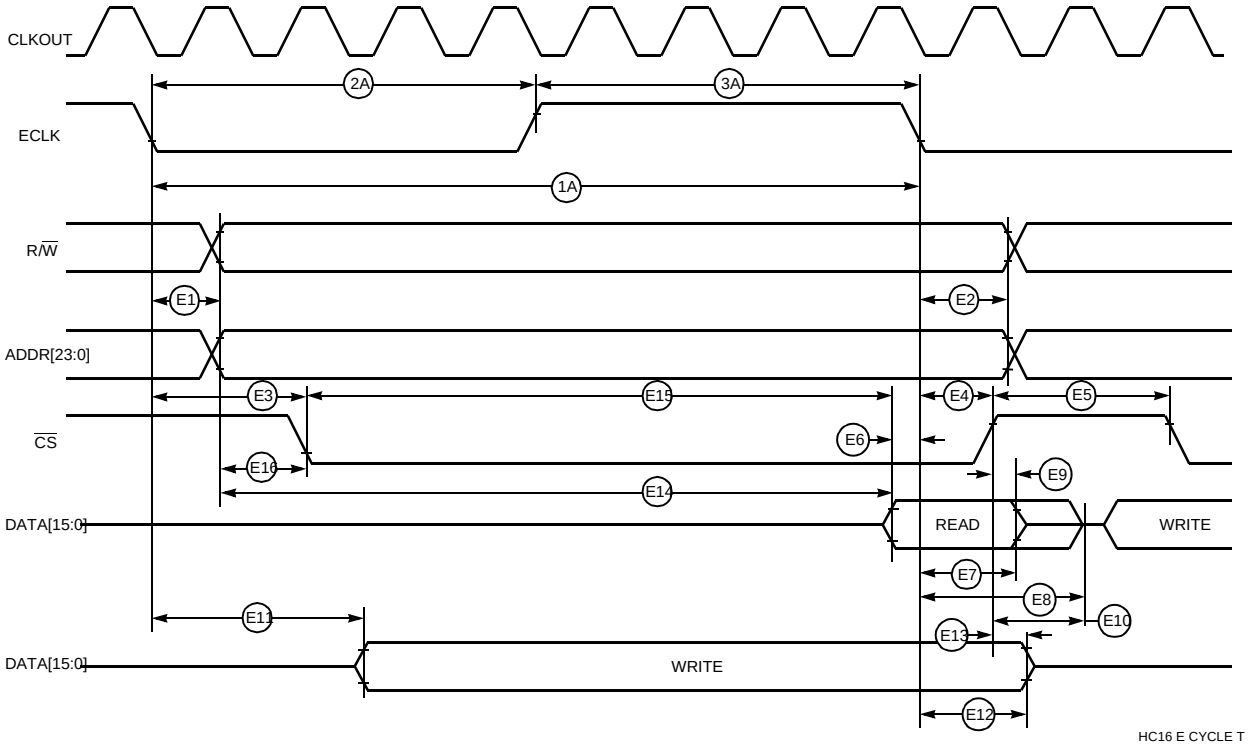


Figure A-15 ECLK Timing Diagram

Table A-9 SPI Timing

(V_{DD} and V_{DDSYN} = 5.0 Vdc ± 5%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H)[1]

Num	Function	Symbol	Min	Max	Unit
	Operating Frequency Master Slave	f_{spi}	DC DC	1/4 1/4	f_{sys} f_{sys}
1	Cycle Time Master Slave	t_{qcyc}	4 4	510 —	t_{cyc} t_{cyc}
2	Enable Lead Time Master Slave	t_{lead}	2 2	128 —	t_{cyc} t_{cyc}
3	Enable Lag Time Master Slave	t_{lag}	— 2	1/2 —	SCK t_{cyc}
4	Clock (SCK) High or Low Time Master Slave[2]	t_{sw}	2 t_{cyc} – 60 2 t_{cyc} – n	255 t_{cyc} —	ns ns
5	Sequential Transfer Delay Master Slave (Does Not Require Deselect)	t_{td}	17 13	8192 —	t_{cyc} t_{cyc}
6	Data Setup Time (Inputs) Master Slave	t_{su}	30 20	— —	ns ns
7	Data Hold Time (Inputs) Master Slave	t_{hi}	0 20	— —	ns ns
8	Slave Access Time	t_a	—	1	t_{cyc}
9	Slave MISO Disable Time	t_{dis}	—	2	t_{cyc}
10	Data Valid (after SCK Edge) Master Slave	t_v	— —	50 50	ns ns
11	Data Hold Time (Outputs) Master Slave	t_{ho}	0 0	— —	ns ns
12	Rise Time Input Output	t_{ri} t_{ro}	— —	2 30	µs ns
13	Fall Time Input Output	t_{fi} t_{fo}	— —	2 30	µs ns

NOTES:
1. All AC timing is shown with respect to V_{IH}/V_{IL} levels unless otherwise noted.
2. For high time, n = External SCK rise time; for low time, n = External SCK fall time.

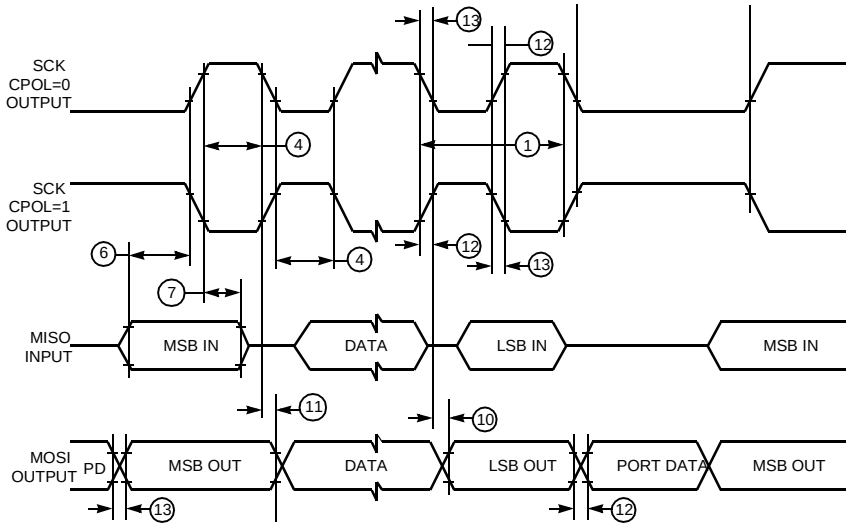


Figure A-16 SPI Timing — Master, CPHA = 0

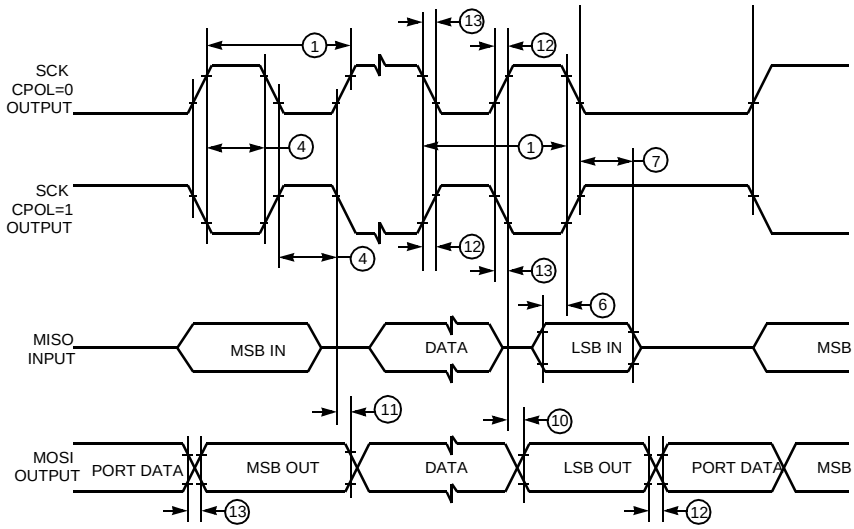


Figure A-17 SPI Timing — Master, CPHA = 1

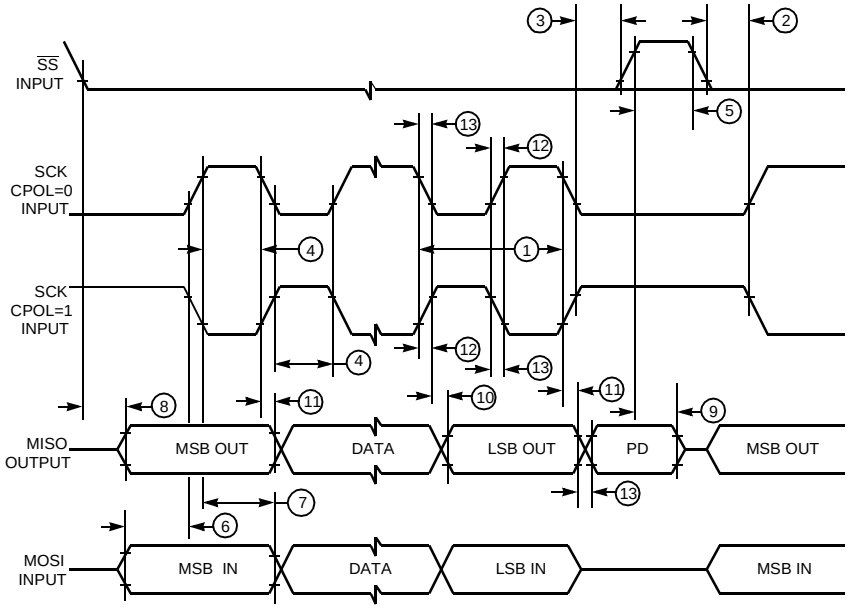


Figure A-18 SPI Timing — Slave, CPHA = 0

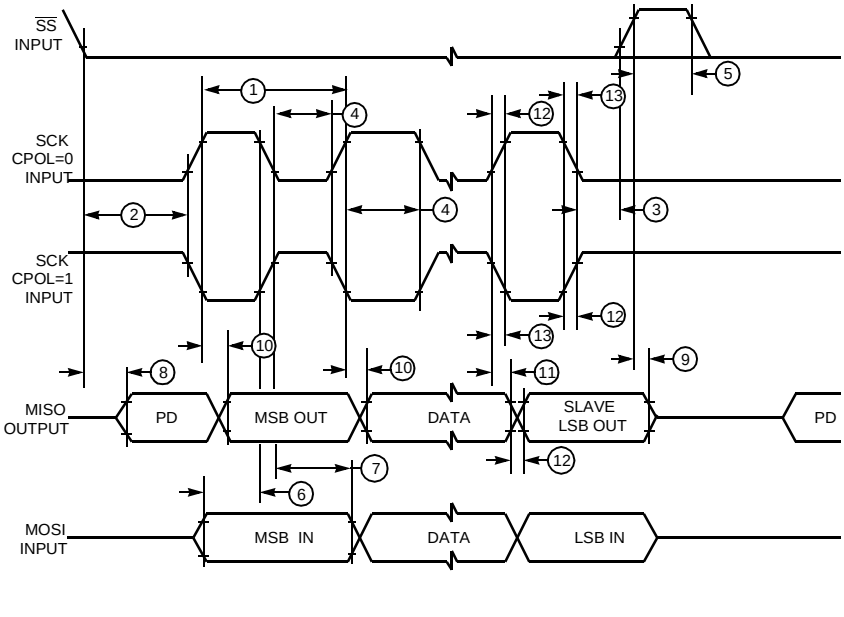


Figure A-19 SPI Timing — Slave, CPHA = 1

Table A-10 ADC Maximum Ratings

Num	Parameter	Symbol	Min	Max	Unit
1	Analog Supply	V_{DDA}	–0.3	6.5	V
2	Internal Digital Supply, with reference to V_{SSI}	V_{DDI}	–0.3	6.5	V
3	Reference Supply, with reference to V_{SSI}	V_{RH}, V_{RL}	–0.3	6.5	V
4	V_{SS} Differential Voltage	$V_{SSI} - V_{SSA}$	–0.1	0.1	V
5	V_{DD} Differential Voltage	$V_{DDI} - V_{DDA}$	–6.5	6.5	V
6	V_{REF} Differential Voltage	$V_{RH} - V_{RL}$	–6.5	6.5	V
7	V_{RH} to V_{DDA} Differential Voltage	$V_{RH} - V_{DDA}$	–6.5	6.5	V
8	V_{RL} to V_{SSA} Differential Voltage	$V_{RL} - V_{SSA}$	–6.5	6.5	V
9	Disruptive Input Current[1, 2, 3, 4, 5, 6, 7] $V_{NEGCLAMP} \cong -0.3$ V $V_{POSCLAMP} \cong 8$ V	I_{NA}	–500	500	µA
10	Positive Overvoltage Current Coupling Ratio[1, 5, 6, 8]	K_P	2000	—	—
11	Negative Overvoltage Current Coupling Ratio[1, 5, 6, 8]	K_N	500	—	—
12	Maximum Input Current [3, 4, 6] $V_{NEGCLAMP} \cong -0.3$ V $V_{POSCLAMP} \cong 8$ V	I_{MA}	–25	25	mA

NOTES:

1. Below disruptive current conditions, a stressed channel will store the maximum conversion value for analog inputs greater than V_{RH} and the minimum conversion value for inputs less than V_{RL}. This assumes that $V_{RH} \leq V_{DDA}$ and $V_{RL} \geq V_{SSA}$ due to the presence of the sample amplifier. Other channels are not affected by non-disruptive conditions
2. Input signals with large slew rates or high frequency noise components cannot be converted accurately. These signals also interfere with conversion of other channels.
3. Exceeding limit may cause conversion error on stressed channels and on unstressed channels. Transitions within the limit do not affect device reliability or cause permanent damage.
4. Input must be current limited to the value specified. To determine the value of the required current-limiting resistor, calculate resistance values using positive and negative clamp values, then use the larger of the calculated values.
5. This parameter is periodically sampled rather than 100% tested.
6. Applies to single pin only.
7. The values of external system components can change the maximum input current value, and affect operation. A voltage drop may occur across the external source impedances of the adjacent pins, impacting conversions on these adjacent pins. The actual maximum may need to be determined by testing the complete design.
8. Current coupling is the ratio of the current induced from overvoltage (positive or negative, through an external series coupling resistor), divided by the current induced on adjacent pins. A voltage drop may occur across the external source impedances of the adjacent pins, impacting conversions on these adjacent pins

Table A-11 ADC DC Electrical Characteristics (Operating)

(V_{SS} = 0 Vdc, ADCLK = 2.1 MHz, $T_A = T_L$ to T_H)

Num	Parameter	Symbol	Min	Max	Unit
1	Analog Supply[1]	V_{DDA}	4.5	5.5	V
2	Internal Digital Supply[1]	V_{DDI}	4.5	5.5	V
3	V_{SS} Differential Voltage	$V_{SSI} - V_{SSA}$	– 1.0	1.0	mV
4	V_{DD} Differential Voltage	$V_{DDI} - V_{DDA}$	– 1.0	1.0	V
5	Reference Voltage Low[2, 3]	V_{RL}	V_{SSA}	V_{DDA} / 2	V
6	Reference Voltage High[2, 3]	V_{RH}	V_{DDA} / 2	V_{DDA}	V
7	V_{REF} Differential Voltage[3]	$V_{RH} - V_{RL}$	4.5	5.5	V
8	Input Voltage[2]	V_{INDC}	V_{SSA}	V_{DDA}	V
9	Input High, Port ADA	V_{IH}	0.7 (V_{DDA})	V_{DDA} + 0.3	V
10	Input Low, Port ADA	V_{IL}	V_{SSA} – 0.3	0.2 (V_{DDA})	V
11	Analog Supply Current Normal Operation[4] Low-power stop	I_{DDA}	— —	1.0 200	mA µA
12	Reference Supply Current	I_{REF}	—	250	µA
13	Input Current, Off Channel[5]	I_{OFF}	—	150	nA
14	Total Input Capacitance, Not Sampling	C_{INN}	—	10	pF
15	Total Input Capacitance, Sampling	C_{INS}	—	15	pF

NOTES:

1. Refers to operation over full temperature and frequency range.
2. To obtain full-scale, full-range results, $V_{SSA} \le V_{RL} \le V_{INDC} \le V_{RH} \le V_{DDA}$.
3. Accuracy tested and guaranteed at $V_{RH} - V_{RL}$ = 5.0 V ± 5%.
4. Current measured at maximum system clock frequency with ADC active.
5. Maximum leakage occurs at maximum operating temperature. Current decreases by approximately one-half for each 10°C decrease from maximum temperature.

Table A-12 ADC AC Characteristics (Operating)

(V_{DD} and V_{DDA} = 5.0 Vdc ± 5%, V_{SS} = 0 Vdc, T_A within operating temperature range)

Num	Parameter	Symbol	Min	Max	Unit
1	ADC Clock Frequency	f_{ADCLK}	0.5	2.1	MHz
2	8-bit Conversion Time[1] f_{ADCLK} = 1.0 MHz f_{ADCLK} = 2.1 MHz	t_{CONV}	15.2 7.6	—	µs
3	10-bit Conversion Time[1] f_{ADCLK} = 1.0 MHz f_{ADCLK} = 2.1 MHz	t_{CONV}	17.1 8.6	—	µs
4	Stop Recovery Time	t_{SR}	—	10	µs

NOTES:

1. Conversion accuracy varies with f_{ADCLK} rate. Reduced conversion accuracy occurs at maximum.

Table A-13 ADC Conversion Characteristics (Operating)

(V_{DD} and V_{DDA} = 5.0 Vdc ± 5%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H,
0.5 MHz ≤ f_{ADCLK} ≤ 1.0 MHz, 2 clock input sample time)

Num	Parameter	Symbol	Min	Typical	Max	Unit
1	8-bit Resolution[1]	1 Count	—	20	—	mV
2	8-bit Differential Nonlinearity	DNL	–0.5	—	0.5	Counts
3	8-bit Integral Nonlinearity	INL	–1	—	1	Counts
4	8-bit Absolute Error[2]	AE	–1	—	1	Counts
5	10-bit Resolution[1]	1 Count	—	5	—	mV
6	10-bit Differential Nonlinearity[3]	DNL	–0.5	—	0.5	Counts
7	10-bit Integral Nonlinearity[3]	INL	–2.0	—	2.0	Counts
8	10-bit Absolute Error[3, 4]	AE	–2.5	—	2.5	Counts
9	Source Impedance at Input[5]	R_S	—	20	—	kΩ

NOTES:

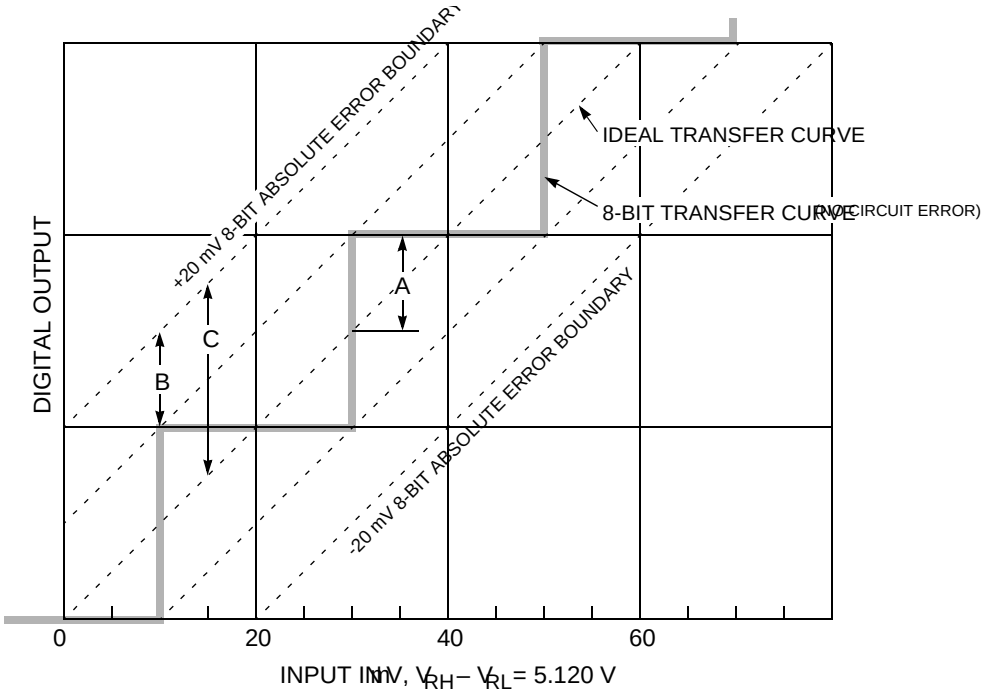
1. At $V_{RH} - V_{RL}$ = 5.12 V, one 10-bit count = 5 mV and one 8-bit count = 20 mV.
2. 8-bit absolute error of 1 count (20 mV) includes 1/2 count (10 mV) inherent quantization error and 1/2 count (10 mV) circuit (differential, integral, and offset) error.
3. Conversion accuracy varies with f_{ADCLK} rate. Reduced conversion accuracy occurs at maximum f_{ADCLK}. Assumes that minimum sample time (2 ADC Clocks) is selected.
4. 10-bit absolute error of 2.5 counts (12.5 mV) includes 1/2 count (2.5 mV) inherent quantization error and 2 counts (10 mV) circuit (differential, integral, and offset) error.
5. Maximum source impedance is application-dependent. Error resulting from pin leakage depends on junction leakage into the pin and on leakage due to charge-sharing with internal capacitance.

   Error from junction leakage is a function of external source impedance and input leakage current. Expected error in result value due to junction leakage is expressed in voltage (V_{ERRJ}):

   $$V_{ERRJ} = R_S \times I_{OFF}$$

   where I_{OFF} is a function of operating temperature, as shown in **Table A-11**.

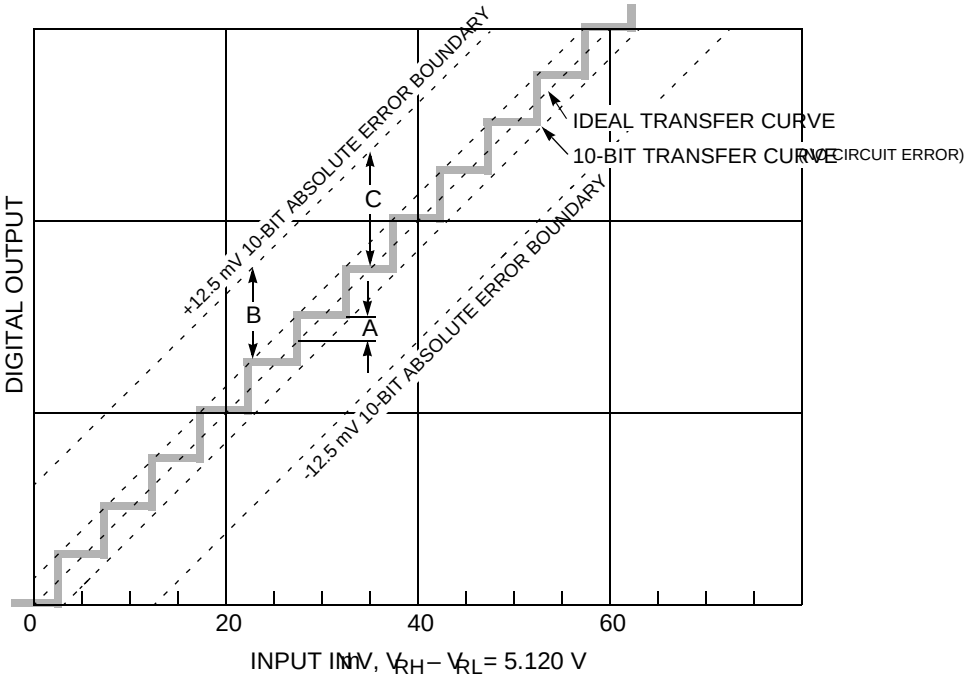
   Charge-sharing leakage is a function of input source impedance, conversion rate, change in voltage between successive conversions, and the size of the decoupling capacitor used. Error levels are best determined empirically. In general, continuous conversion of the same channel may not be compatible with high source impedance.



A – +1/2 COUNT (10 mV) INHERENT QUANTIZATION ERROR
B – CIRCUIT-CONTRIBUTED ERROR
C – + 20 mV ABSOLUTE ERROR (ONE 8-BIT COUNT)

ADC 8-BIT ACCURAC

Figure A-20 8-Bit ADC Conversion Accuracy



A – +.5 COUNT (2.5 mV) INHERENT QUANTIZATION ERROR
B – CIRCUIT-CONTRIBUTED 1 mV ERROR
C – +12.5 mV ABSOLUTE ERROR (2.5 10-BIT COUNTS)

ADC 10-BIT ACCURACY

Figure A-21 10-Bit ADC Conversion Accuracy

Table A-14 BEFLASH/Flash EEPROM Module Specifications

Num	Characteristic	Symbol	Min	Max	Unit
1	Program/Erase Supply Voltage[1] Read Operation Program/Erase/Verify Operation	V_{FPE}	 V_{DD} – 0.5 11.4	 5.5 12.6	V
2	Program/Erase Supply Current[2] Read Operation Program/Erase/Verify Operation Verify (ENPE = 0) Program Byte (ENPE = 1) Program Word (ENPE = 1) Erase (ENPE = 1)	I_{FPE}	 — — — — —	 15 50 15 30 4	 µA µA mA mA mA
3	Program Recovery Time[3]	t_{pr}	1	—	µsecs
4	Program Pulse Width	pw_{pp}	20	25	µsecs
5	Number of Program Pulses[4]	n_{pp}	—	50	—
6	Program Margin[5]	p_m	100	—	%
7	Number of Erase Pulses[4]	n_{ep}	—	5	—
8	Erase Pulse Time ($t_{ei} \times k$)	t_{epk}	90	550	ms
9	Amount to Increment t_{ep}	t_{ei}	90	110	ms
10	Erase Margin $\sum_{k=1}^{n_{ep}} t_{ei} \times k$	e_m	90	1650	ms
11	Erase Recovery Time[3]	t_{er}	—	1	ms
12	Low-Power Stop Recovery Time[3, 6]	t_{sb}	—	1	µsecs

NOTES:

1. V_{FPE} must not be raised to programming voltage while V_{DD} is below specified minimum value. V_{FPE} must not be reduced below minimum specified value while V_{DD} is applied.
2. Current parameters apply to each individual EEPROM module.
3. Minimum software delay from the end of the write cycle that clears ENPE bit to the read of the flash array
4. Without margin.
5. At 100% margin, the number of margin pulses required is the same as the number of pulses used to program the byte or word.
6. Minimum software delay from the end of the write cycle that clears the STOP bit to the read of the flash array.

Table A-15 BEFLASH/Flash EEPROM Module Life

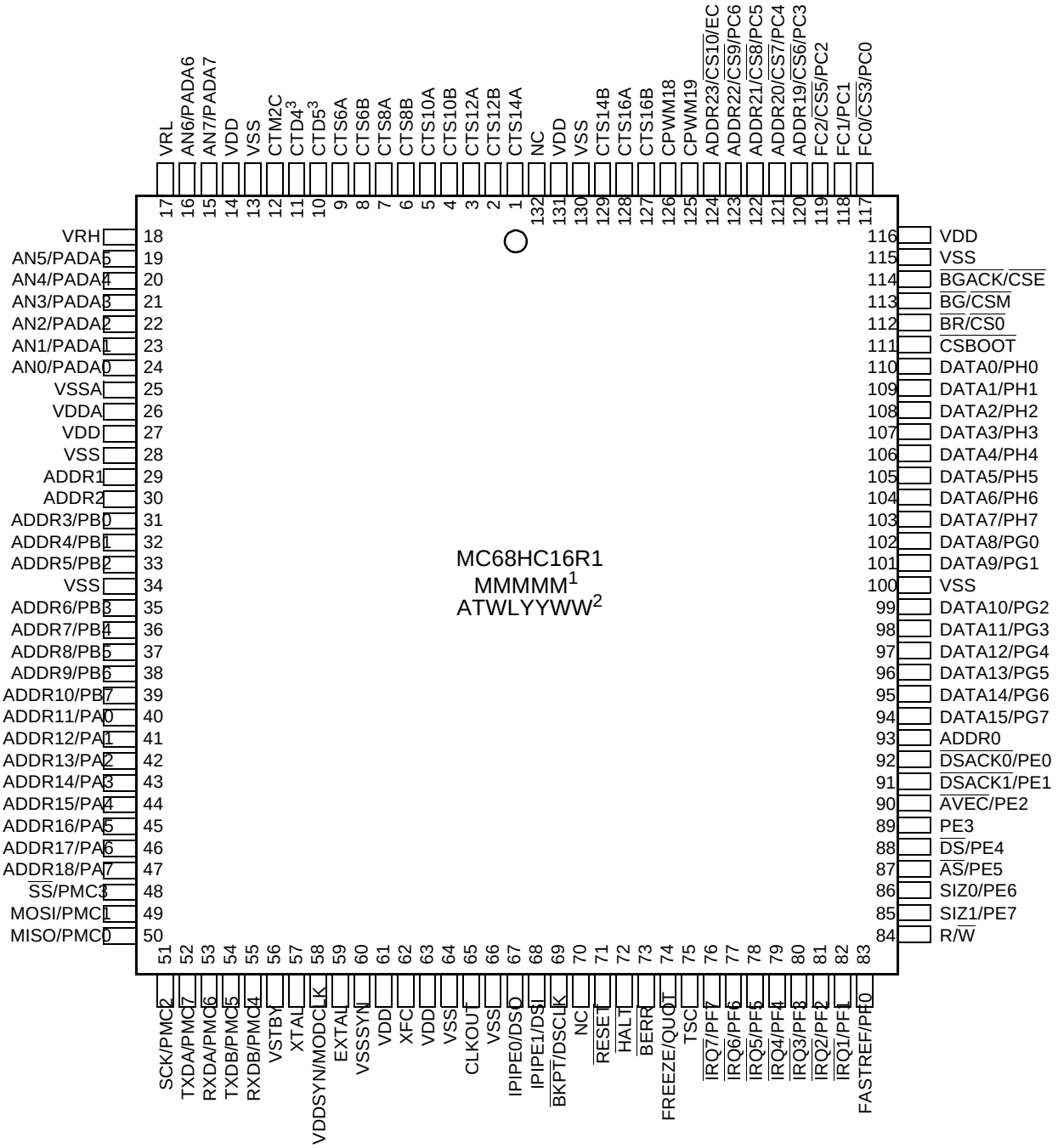
Num	Parameter	Symbol	Value	Unit
1	Program-Erase Endurance[1]	e_{pe}	100	cyc
2	Data Retention[2]	r_d	10	yr

NOTES:

1. Number of program-erase cycles (1 to 0, 0 to 1) per bit.
2. Parameter based on accelerated-life testing with standard test pattern.

APPENDIX B
MECHANICAL DATA AND ORDERING INFORMATION

M68HC16 R-series microcontrollers are available in a 132-pin plastic surface mount package. This appendix provides package pin assignment drawings, a dimensional drawing, and instructions on obtaining updated mechanical data and ordering information.

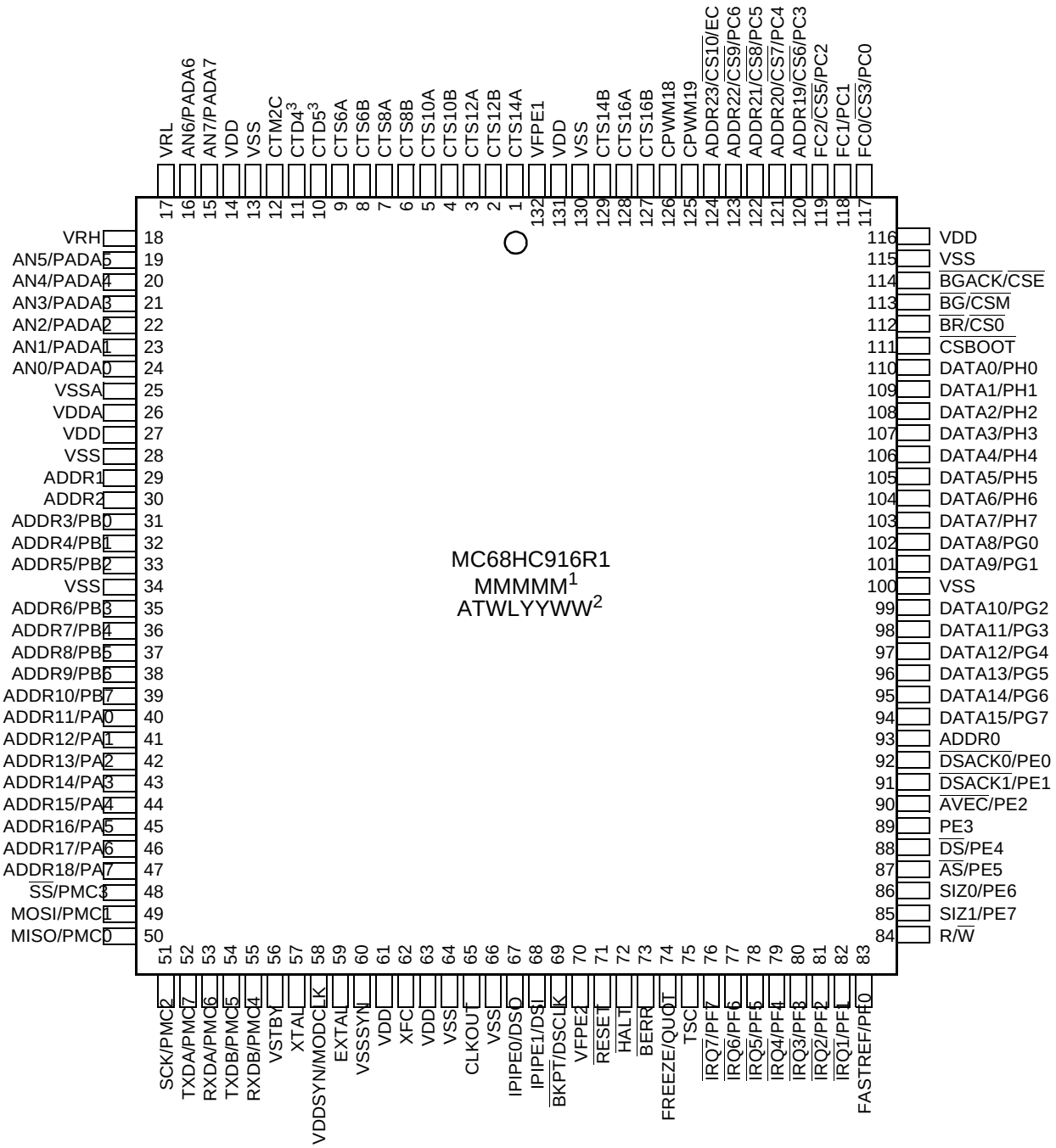


NOTES:

1. MMMMM = MASK OPTION NUMBER
2. ATWLYYWW = ASSEMBLY TEST LOCATION/YEAR, WEEK
3. R-SERIES MCUS ARE COMPATIBLE. THE ONLY DIFFERENCE IS IN THE NAMING OF THE CTM7 AND CTM8 DOUBLE-ACTION SUBMODULE PINS.

MC68HC16R1 132-PIN QF

Figure B-1 MC68HC16R1 Pin Assignment for 132-Pin Package

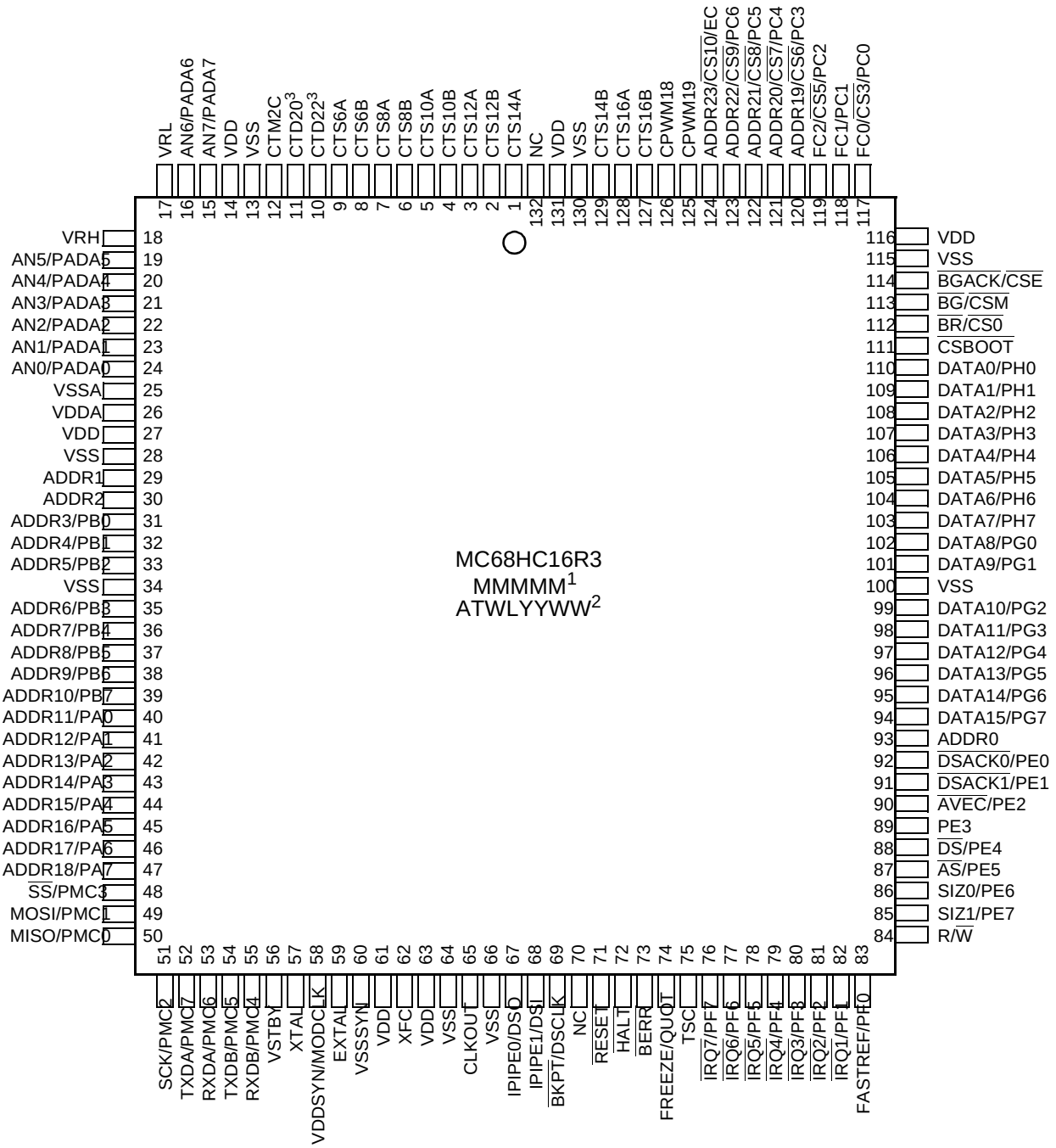


NOTES:

1. MMMMM = MASK OPTION NUMBER
2. ATWLYYWW = ASSEMBLY TEST LOCATION/YEAR, WEEK
3. R-SERIES MCUS ARE COMPATIBLE. THE ONLY DIFFERENCE IS IN THE NAMING OF THE CTM7 AND CTM8 DOUBLE-ACTION SUBMODULE PINS.

MC68HC916R1 132-PIN QF

Figure B-2 MC68HC916R1 Pin Assignment for 132-Pin Package

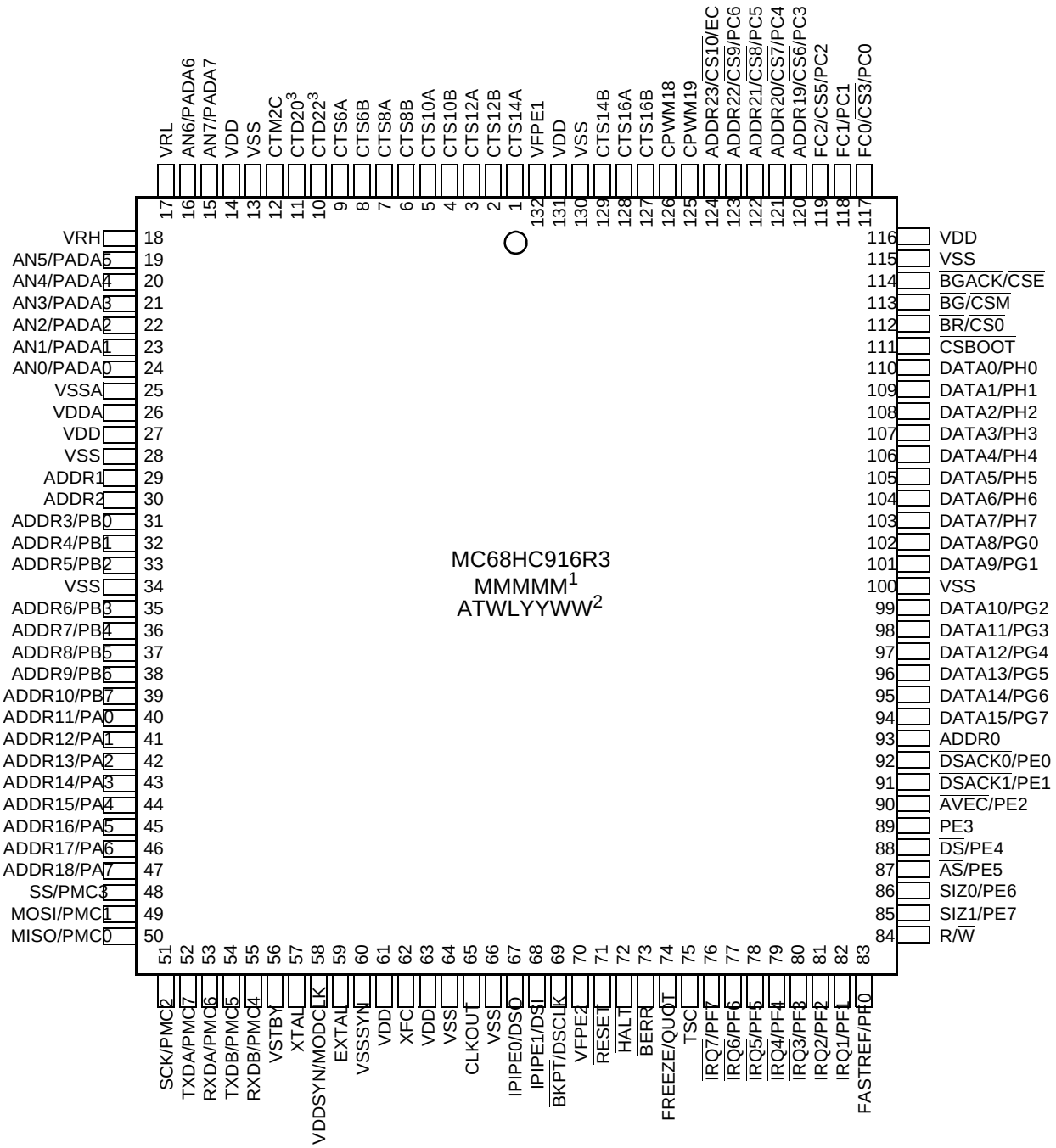


NOTES:

1. MMMMM = MASK OPTION NUMBER
2. ATWLYYWW = ASSEMBLY TEST LOCATION/YEAR, WEEK
3. R-SERIES MCUS ARE COMPATIBLE. THE ONLY DIFFERENCE IS IN THE NAMING OF THE CTM7 AND CTM8 DOUBLE-ACTION SUBMODULE PINS.

MC68HC16R3 132-PIN QF

Figure B-3 MC68HC16R3 Pin Assignment for 132-Pin Package

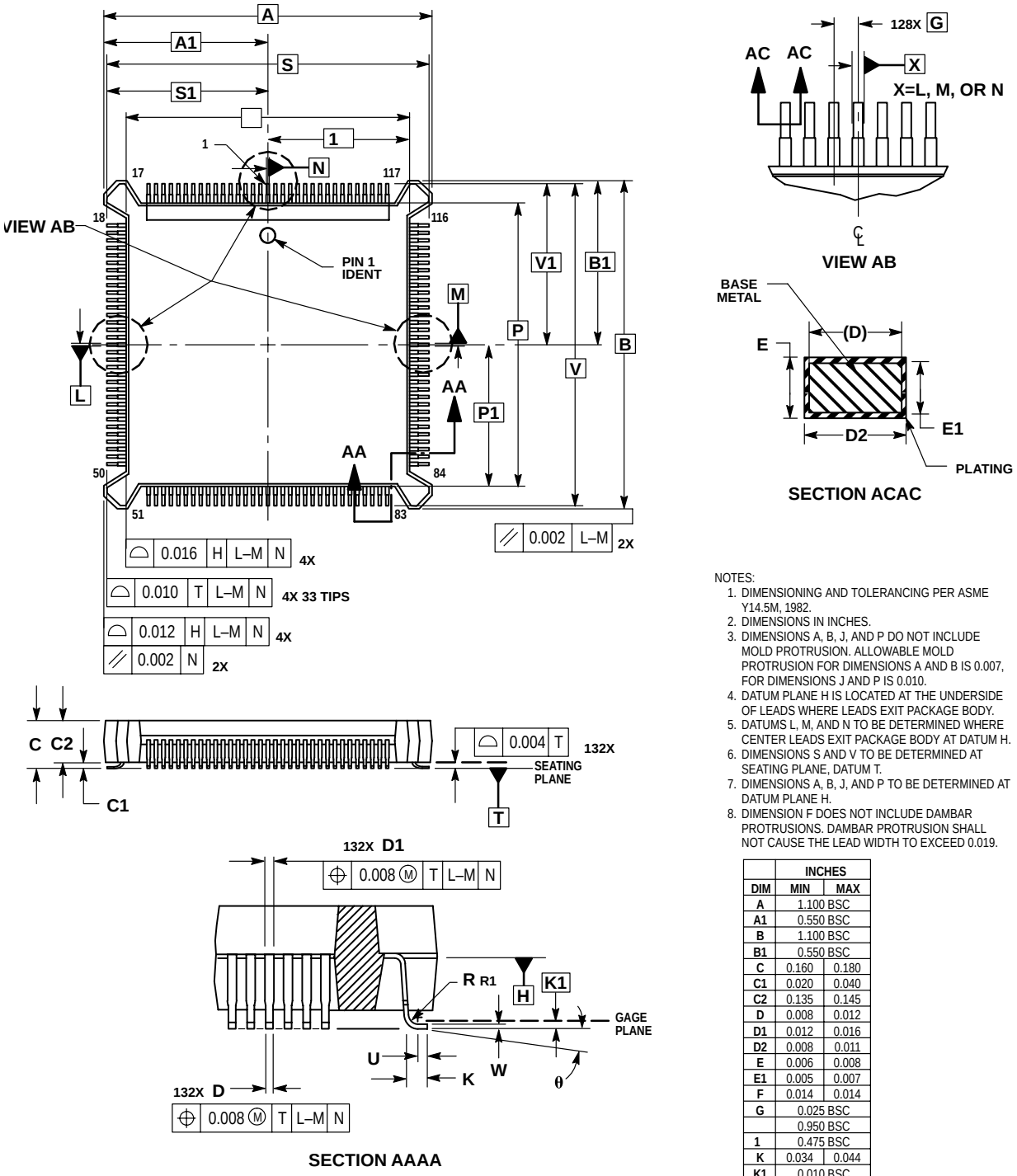


NOTES:

1. MMMMM = MASK OPTION NUMBER
2. ATWLYYWW = ASSEMBLY TEST LOCATION/YEAR, WEEK
3. R-SERIES MCUS ARE COMPATIBLE. THE ONLY DIFFERENCE IS IN THE NAMING OF THE CTM7 AND CTM8 DOUBLE-ACTION SUBMODULE PINS.

MC68HC916R3 132-PIN QF

Figure B-4 MC68HC916R3 Pin Assignment for 132-Pin Package



NOTES:
1. DIMENSIONING AND TOLERANCING PER ASME Y14.5M, 1982.
2. DIMENSIONS IN INCHES.
3. DIMENSIONS A, B, J, AND P DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE MOLD PROTRUSION FOR DIMENSIONS A AND B IS 0.007, FOR DIMENSIONS J AND P IS 0.010.
4. DATUM PLANE H IS LOCATED AT THE UNDERSIDE OF LEADS WHERE LEADS EXIT PACKAGE BODY.
5. DATUMS L, M, AND N TO BE DETERMINED WHERE CENTER LEADS EXIT PACKAGE BODY AT DATUM H.
6. DIMENSIONS S AND V TO BE DETERMINED AT SEATING PLANE, DATUM T.
7. DIMENSIONS A, B, J, AND P TO BE DETERMINED AT DATUM PLANE H.
8. DIMENSION F DOES NOT INCLUDE DAMBAR PROTRUSIONS. DAMBAR PROTRUSION SHALL NOT CAUSE THE LEAD WIDTH TO EXCEED 0.019.

	INCHES	
DIM	MIN	MAX
A	1.100 BSC	
A1	0.550 BSC	
B	1.100 BSC	
B1	0.550 BSC	
C	0.160	0.180
C1	0.020	0.040
C2	0.135	0.145
D	0.008	0.012
D1	0.012	0.016
D2	0.008	0.011
E	0.006	0.008
E1	0.005	0.007
F	0.014	0.014
G	0.025 BSC	
	0.950 BSC	
1	0.475 BSC	
K	0.034	0.044
K1	0.010 BSC	
P	0.950 BSC	
P1	0.475 BSC	
R1	0.013 REF	
S	1.080 BSC	
S1	0.540 BSC	
U	0.025 REF	
V	1.080 BSC	
V1	0.540 BSC	
W	0.006	0.008
θ	0 °	8 °

Figure B-5 Case 831A-01 — 132-Pin Package Dimensions

B.1 Obtaining Updated M68HC16 R-Series MCU Mechanical Information

Although all devices manufactured by Motorola conform to current JEDEC standards, complete mechanical information regarding M68HC16 R-series microcontrollers is available through Motorola's Design-Net.

To download updated package specifications, perform the following steps:

1. Visit the Design-Net case outline database search engine at http://design-net.com/cgi-bin/cases.
2. Enter the case outline number, located in **Figure B-5** without the revision code (for example, 831A, not 831A-01) in the field next to the search button.
3. Download the file with the new package diagram.

B.2 Obtaining Ordering Information

Refer to the current product selector guide (SG166/G) for current part availability.

APPENDIX C
DEVELOPMENT SUPPORT

This section serves as a brief reference to Motorola development tools for M68HC16 R-series microcontrollers.

Information provided is complete as of the time of publication, but new systems and software are continually being developed. In addition, there is a growing number of third-party tools available. The Motorola *Microcontroller Development Tools Directory* (MCUDEVTLDIR/D Revision. 3) provides an up-to-date list of development tools. Contact your Motorola representative for further information.

C.1 M68MMDS1632 Modular Development System

The M68MMDS1632 Motorola Modular Development System (MMDS) is a development tool for evaluating M68HC16 and M68300 MCU-based systems. The MMDS1632 is an in-circuit emulator, which includes a station module and active probe. A separately purchased MPB and PPB completes MMDS functionality with regard to a particular MCU or MCU family. The many MPBs and PPBs available let your MMDS emulate a variety of different MCUs. Contact your Motorola sales representative, who will assist you in selecting and configuring the modular system that fits your needs. A full-featured development system, the MMDS provides both in-circuit emulation and bus analysis capabilities, including:

- Real-time in-circuit emulation at maximum speed of 16 MHz
- Built-in emulation memory
  - — 1-Mbyte main emulation memory (three-clock bus cycle)
  - — 256-Kbyte fast termination (two-clock bus cycle)
  - — 4-Kbyte dual-port emulation memory (three-clock bus cycle)
- Real-time bus analysis
  - — Instruction disassembly
  - — State-machine-controlled triggering
- Four hardware breakpoints, bitwise masking
- Analog/digital emulation
- Synchronized signal output
- Built-in AC power supply, 90 - 264 V, 50 - 60 Hz, FCC and EC EMI compliant
- RS-232 connection to host capable of communicating at 1200, 2400, 4800, 9600, 19200, 38400, or 57600 baud

C.2 M68MEVB1632 Modular Evaluation Board

The M68MEVB1632 Modular Evaluation Board (MEVB) is a development tool for evaluating M68HC16 and M68300 MCU-based systems. The MEVB consists of the M68MPFB1632 modular platform board, an MCU personality board (MPB), an in-circuit debugger (ICD16 or ICD32), and development software. MEVB features include:

- An economical means of evaluating target systems incorporating M68HC16 and M68300 HCMOS MCU devices.
- Expansion memory sockets for installing RAM, EPROM, or EEPROM.
  - — Data RAM: 32K x 16, 128K x 16, or 512K x 16
  - — EPROM/EEPROM: 32K x 16, 64K x 16, 128K x 16, 256K x 16, or 512K x 16
  - — Fast RAM: 32K x 16 or 128K x 16
- Background-mode operation, for detailed operation from a personal computer platform without an on-board monitor.
- Integrated assembly/editing/evaluation/programming environment for easy development.
- As many as seven software breakpoints.
- Re-usable ICD hardware for your target application debug or control.
- Two RS-232C terminal input/output (I/O) ports for user evaluation of the serial communication interface.
- Logic analyzer pod connectors.
- Port replacement unit (PRU) to rebuild I/O ports lost to address/data/control.
- On-board V_{PP} (+12 VDC) generation for MCU and flash EEPROM programming.
- On-board wire-wrap area.

APPENDIX D
REGISTER SUMMARY

This appendix contains address maps, register diagrams, and bit/field definitions for M68HC16 R-series MCUs. More detailed information about register function is provided in the appropriate sections of the manual.

Except for central processing unit resources, information is presented in the intermodule bus address order shown in **Table D-1**.

Table D-1 Module Address Map

Module	Size (Bytes)	Base Address
SCIM2	128	$YFFA00
2K SRAM (MC68HC16R1/MC68HC916R1only)	8	$YFFB00
4K SRAM (MC68HC16R3/MC68HC916R3only)	8	$YFFB00
48K MRM (MC68HC16R1 only)	32	$YFF820
96K (32K + 64K) MRM (MC68HC16R3 only)	32	$YFF820
ADC	64	$YFF700
MCCI	64	$YFFC00
CTM7 (MC68HC16R1/MC68HC916R1only)	256	$YFF900
CTM8 (MC68HC16R3/MC68HC916R3only)	256	$YFF900
48K (16K + 32K) FLASH EEPROM (MC68HC916R1 only)	32	$YFF800
96K (32K X 3) FLASH EEPROMs (MC68HC916R3 only)	32	$YFF800
BEFLASH (MC68HC916R1/MC68HC916R3 only)	2	$YFF7A0

Control registers for all the modules in the microcontroller are mapped into a 4-Kbyte block. The state of the module mapping (MM) bit in SCIMCR determines where the control registers block is located in the system memory map. When MM = 0, register addresses range from $7FF000 to $7FFFFF; when MM = 1, register addresses range from $FFF000 to $FFFFFF.

With the CPU16, ADDR[23:20] follow the logic state of ADDR19 unless driven externally. MM corresponds to IMB ADDR23. If it is cleared, the SCIM2 maps IMB modules into address space $7FF000 – $7FFFFF, which is inaccessible to the CPU16. Modules remain inaccessible until reset occurs. The reset state of MM is one, but the bit is onetime writable. Initialization software should make certain it remains set.

D.1 Central Processing Unit

CPU16 registers are not part of the module address map. **Figure D-1** is a functional representation of CPU resources.

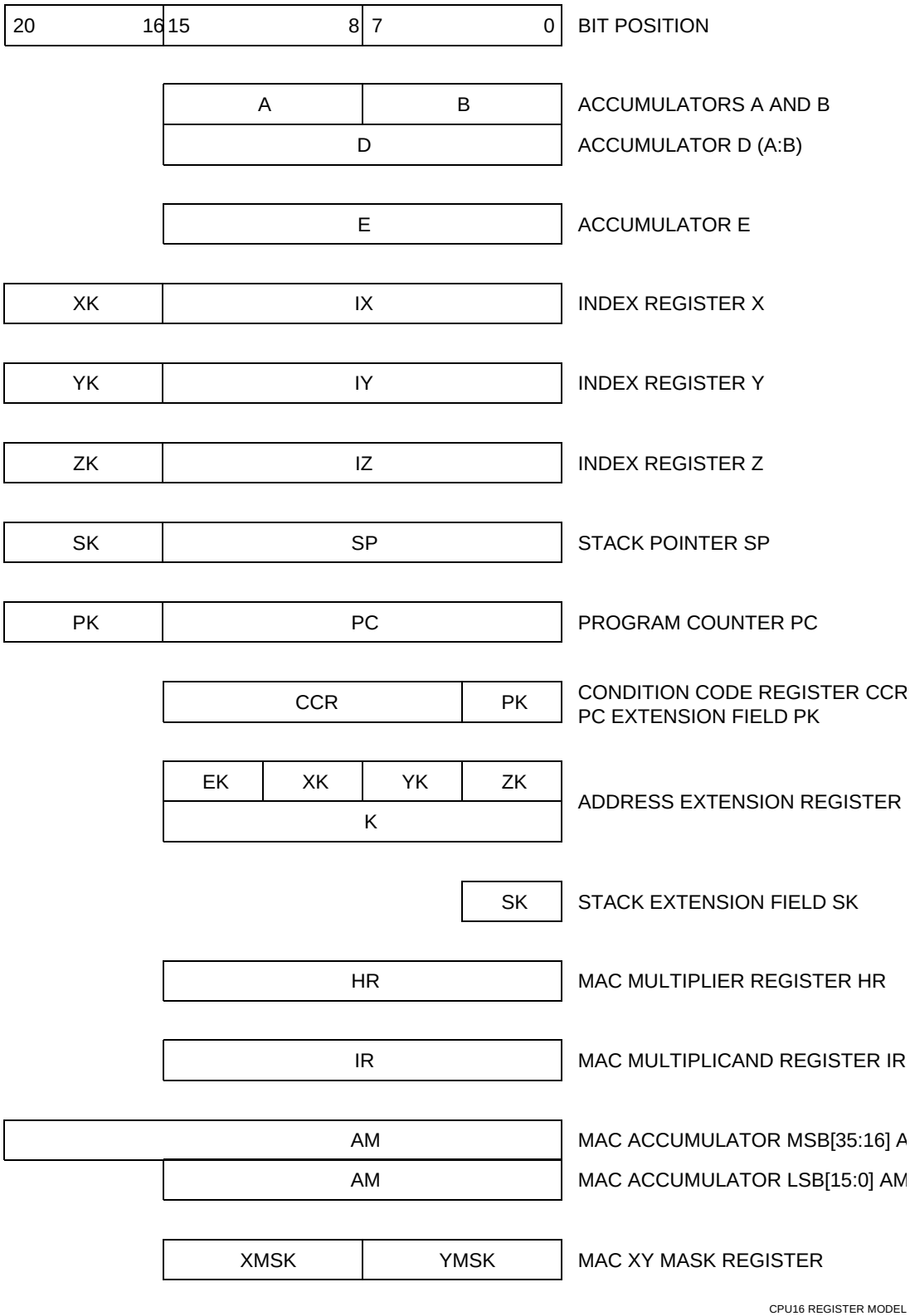


Figure D-1 CPU16 Register Model

D.1.1 Condition Code Register

CCR — Condition Code Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S	MV	H	EV	N	Z	V	C	IP[2:0]			SM	PK[3:0]			

The CCR contains processor status flags, the interrupt priority field, and the program counter address extension field. The CPU16 has a special set of instructions that manipulate the CCR.

S — STOP Enable
0 = Stop CPU16 clocks when LPSTOP instruction is executed.
1 = Perform NOPs when LPSTOP instruction is executed.

MV — Accumulator M overflow flag
Set when overflow into AM35 has occurred.

H — Half Carry Flag
Set when a carry from A3 or B3 occurs during BCD addition.

EV — Extension Bit Overflow Flag
Set when an overflow into AM31 has occurred.

N — Negative Flag
Set when the MSB of a result register is set.

Z — Zero Flag
Set when all bits of a result register are zero.

V — Overflow Flag
Set when two’s complement overflow occurs as the result of an operation.

C — Carry Flag
Set when carry or borrow occurs during arithmetic operation. Also used during shifts and rotates.

IP[2:0] — Interrupt Priority Field
The priority value in this field (0 to 7) is used to mask low priority interrupts.

SM — Saturate Mode Bit
When SM is set, if either EV or MV is set, data read from AM using TMER or TMET will be given maximum positive or negative value, depending on the state of the AM sign bit before overflow occurred.

PK[3:0] — Program Counter Address Extension Field
This field is concatenated with the program counter to form a 20-bit address. There are no instructions to manipulate this field. The CPU16 updates the PK field automatically and transparently to the user.

D.2 Single-Chip Integration Module 2

Table D-2 shows the SCIM2 address map.

Table D-2 SCIM2 Address Map

Address[1]	15 8	7 0
$YFFA00	SCIM Module Configuration Register (SCIMCR)	
$YFFA02	SCIM Test Register (SCIMTR)	
$YFFA04	Clock Synthesizer Control Register (SYNCR)	
$YFFA06	Not Used	Reset Status Register (RSR)
$YFFA08	SCIM Test Register E (SCIMTRE)	
$YFFA0A	Port A Data Register (PORTA)	Port B Data Register (PORTB)
$YFFA0C	Port G Data Register (PORTG)	Port H Data Register (PORTH)
$YFFA0E	Port G Data Direction Register (DDRG)	Port H Data Direction Register (DDRH)
$YFFA10	Not Used	Port E Data Register 0(PORTE0)
$YFFA12	Not Used	Port E Data Register 1(PORTE1)
$YFFA14	Port A/B Data Direction Register (DDRAB)	Port E Data Direction Register (DDRE)
$YFFA16	Not Used	Port E Pin Assignment Register (PEPAR)
$YFFA18	Not Used	Port F Data Register 0 (PORTF0)
$YFFA1A	Not Used	Port F Data Register 1 (PORTF1)
$YFFA1C	Not Used	Port F Data Direction Register (DDRF)
$YFFA1E	Not Used	Port F Pin Assignment Register (PFPAR)
$YFFA20	Not Used	System Protection Control Register (SYPCR)
$YFFA22	Periodic Interrupt Control Register (PICR)	
$YFFA24	Periodic Interrupt Timing Register (PITR)	
$YFFA26	Not Used	Software Service Register (SWSR)
$YFFA28	Not Used	Port F Edge-Detect Flags (PORTFE)
$YFFA2A	Not Used	Port F Edge-Detect Interrupt Vector (PFIVR)
$YFFA2C	Not Used	Port F Edge-Detect Interrupt Level (PFLVR)
$YFFA2E	Not Used	
$YFFA30	Test Module Master Shift A Register (TSTMSRA)	
$YFFA32	Test Module Master Shift B Register (TSTMSRB)	
$YFFA34	Test Module Shift Count Register (TSTSC)	
$YFFA36	Test Module Repetition Counter Register (TSTRC)	
$YFFA38	Test Module Control Register (CREG)	
$YFFA3A	Test Module Distributed Register (DREG)	
$YFFA3C	Not Used	
$YFFA3E	Not Used	
$YFFA40	Not Used	Port C Data Register (PORTC)
$YFFA42	Not Used	Not Used
$YFFA44	Chip-Select Pin Assignment Register 0 (CSPAR0)	
$YFFA46	Chip-Select Pin Assignment Register 1 (CSPAR1)	
$YFFA48	Chip-Select Base Address Register Boot (CSBARBT)	
$YFFA4A	Chip-Select Option Register Boot (CSORBT)	
$YFFA4C	Chip-Select Base Address Register 0 (CSBAR0)	

Table D-2 SCIM2 Address Map (Continued)

Address[1]	15 8	7 0
$YFFA4E	Chip-Select Option Address Register 0 (CSOR0)	
$YFFA50	Not Used	
$YFFA52	Not Used	
$YFFA54	Not Used	
$YFFA56	Not Used	
$YFFA58	Chip-Select Base Address Register 3 (CSBAR3)	
$YFFA5A	Chip-Select Option Address Register 3 (CSOR3)	
$YFFA5C	Not Used	
$YFFA5E	Not Used	
$YFFA60	Chip-Select Base Address Register 5 (CSBAR5)	
$YFFA62	Chip-Select Option Address Register 5 (CSOR5)	
$YFFA64	Chip-Select Base Address Register 6 (CSBAR6)	
$YFFA66	Chip-Select Option Address Register 6 (CSOR6)	
$YFFA68	Chip-Select Base Address Register 7 (CSBAR7)	
$YFFA6A	Chip-Select Option Address Register 7 (CSOR7)	
$YFFA6C	Chip-Select Base Address Register 8 (CSBAR8)	
$YFFA6E	Chip-Select Option Address Register 8 (CSOR8)	
$YFFA70	Chip-Select Base Address Register 9 (CSBAR9)	
$YFFA72	Chip-Select Option Address Register 9 (CSOR9)	
$YFFA74	Chip-Select Base Address Register 10 (CSBAR10)	
$YFFA76	Chip-Select Option Address Register 10 (CSOR10)	
$YFFA78	Not Used	
$YFFA7A	Not Used	
$YFFA7C	Not Used	
$YFFA7E	Not Used	

NOTES:
1. Y = M111, where M is the logic state of the module mapping (MM) bit in the SCIMCR.

D.2.1 SCIM Configuration Register

SCIMCR — SCIM Module Configuration Register **$YFFA00**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXOFF	FRZSW	FRZBM	CPUD[1]	RSVD[2]	0	SHEN		SUPV	MM	ABD[1]	RWD[1]	IARB			
RESET:															
0	1	1	*	0	0	0	0	1	1	*	*	1	1	1	1

NOTES:
1. Reset state is mode-dependent. Refer to the following bit descriptions.
2. This bit is reserved for future use. Ensure that initialization software does not change its value (it should always read zero).

SCIMCR controls system configuration. SCIMCR can be read or written at any time, except for the module mapping (MM) bit, which can only be written once after reset, and the reserved bit, which is read-only. Write has no effect.

EXOFF — External Clock Off

0 = The CLKOUT pin is driven during normal operation.

1 = The CLKOUT pin is placed in a high-impedance state.

FRZSW — Freeze Software Enable

0 = When FREEZE is asserted, the software watchdog and periodic interrupt timer continue to operate, allowing interrupts during background debug mode.

1 = When FREEZE is asserted, the software watchdog and periodic interrupt timer are disabled, preventing interrupts during background debug mode.

CPUD — CPU Development Support Disable

0 = Instruction pipeline signals available on pins IPIPE1 and IPIPE0.

1 = Pins IPIPE1 and IPIPE0 placed in high-impedance state unless a breakpoint occurs.

CPUD is cleared to zero when the MCU is in an expanded mode, and set to one in single-chip mode.

FRZBM — Freeze Bus Monitor Enable

0 = When FREEZE is asserted, the bus monitor continues to operate.

1 = When FREEZE is asserted, the bus monitor is disabled.

SHEN[1:0] — Show Cycle Enable

The SHEN field determines how the external bus is driven during internal transfer operations. A show cycle allows internal transfers to be monitored externally.

Table D-3 indicates whether show cycle data is driven externally, and whether external bus arbitration can occur. To prevent bus conflict, external devices must not be selected during show cycles.

Table D-3 Show Cycle Enable Bits

SHEN[1:0]	Effect
00	Show cycles disabled, external arbitration enabled
01	Show cycles enabled, external arbitration disabled
10	Show cycles enabled, external arbitration enabled
11	Show cycles enabled, external arbitration enabled; internal activity is halted by a bus grant

SUPV — Supervisor/User Data Space

This bit has no effect because the CPU16 always operates in the supervisor mode.

MM — Module Mapping

0 = Internal modules are addressed from $7FF000 – $7FFFFF.

1 = Internal modules are addressed from $FFF000 – $FFFFFF.

The logic state of the MM determines the value of ADDR23 for IMB module addresses. Because ADDR[23:20] are driven to the same state as ADDR19, MM must be set to one. If MM is cleared, IMB modules are inaccessible to the CPU16. This bit can be written only once after reset.

ABD — Address Bus Disable

0 = Pins ADDR[2:0] operate normally.

1 = Pins ADDR[2:0] are disabled.

ABD is cleared to zero when the MCU is in an expanded mode, and set to one in single-chip mode. ABD can be written only once after reset.

RWD — Read/Write Disable

0 = R/$\overline{W}$ signal operates normally

1 = R/$\overline{W}$ signal placed in high-impedance state.

RWD is cleared to zero when the MCU is in an expanded mode, and set to one in single-chip mode. RWD can be written only once after reset.

IARB[3:0] — Interrupt Arbitration ID

Each module that can generate interrupts, including the SCIM2, has an IARB field. Each IARB field can be assigned a value from $0 to $F. During an interrupt acknowledge cycle, IARB permits arbitration among simultaneous interrupts of the same priority level. The reset value of the SCIM2 IARB field is $F, the highest priority. This prevents SCIM2 interrupts from being discarded during system initialization.

D.2.2 SCIM Test Register

SCIMTR — Single-Chip Integration Module Test Register **$YFFA02**

Used for factory test only.

D.2.3 Clock Synthesizer Control Register

SYNCR — Clock Synthesizer Control Register **$YFFA04**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W	X	Y[5:0]						EDIV	0	0	RESERVED[1]	SLOCK	RESERVED[1]	STSCIM	STEXT
RESET:															
0	0	1	1	1	1	1	1	0	0	0	0	U	0	0	0

NOTES:

1. Ensure that initialization software does not change the value of these bits. They should always be 0.

This register determines system clock operating frequency and operation during low-power stop mode. With a slow reference frequency between 25 and 50 kHz (typically a 32.768-kHz crystal), the clock frequency is determined by the following equation:

$$f_{sys} = 4f_{ref}(Y + 1)(2^{(2W + X)})$$

With a fast reference frequency between 1 and 6 MHz (typically a 4.194-MHz crystal), the reference frequency is divided by 128 before it is passed to the PLL system. The clock frequency is determined by the following equation:

$$f_{sys} = \frac{f_{ref}(W + 1)}{(2 - X)(2^{Y})}, \; Y \leq 6$$

W — Frequency Control (VCO)

This bit controls a prescaler tap in the synthesizer feedback loop. Setting this bit increases the VCO speed by a factor of four. VCO relock delay is required.

X — Frequency Control (Prescaler)

This bit controls a divide by two prescaler that is not in the synthesizer feedback loop. Setting the bit doubles clock speed without changing the VCO speed. No VCO relock delay is required.

Y[5:0] — Frequency Control (Counter)

The Y field controls the modulus down counter in the synthesizer feedback loop, causing it to divide by a value of Y + 1. Values range from 0 to 63. VCO relock delay is required.

EDIV — E Clock Divide Rate

0 = ECLK frequency is system clock divided by 8.
1 = ECLK frequency is system clock divided by 16.

SLOCK — Synthesizer Lock Flag

0 = VCO is enabled, but has not locked.
1 = VCO has locked on the desired frequency or VCO is disabled.

The MCU remains in reset until the synthesizer locks, but SLOCK does not indicate synthesizer lock status until after the user writes to SYNCR.

STSCIM — Stop Mode SCIM Clock

0 = When LPSTOP is executed, the SCIM clock is driven from the external crystal oscillator and the VCO is turned off to conserve power.
1 = When LPSTOP is executed, the SCIM clock is driven from the internal VCO.

STEXT — Stop Mode External Clock

0 = When LPSTOP is executed, the CLKOUT signal is held negated to conserve power.
1 = When LPSTOP is executed and EXOFF ¦ 1 in SCIMCR, the CLKOUT signal is driven from the SCIM2 clock, as determined by the state of the STSCIM bit.

D.2.4 Reset Status Register

RSR — Reset Status Register **$YFFA06**

15 … 8	7	6	5	4	3	2	1	0
NOT USED	EXT	POW	SW	HLT	0	RE-SERVED	SYS	TST

RSR contains a status bit for each reset source in the MCU. RSR is updated when the MCU comes out of reset. A set bit indicates what type of reset occurred. If multiple sources assert reset signals at the same time, more than one bit in RSR may be set. This register can be read at any time; a write has no effect. Bits [15:8] are unimplemented and always read zero.

EXT — External Reset
Reset caused by the $\overline{\text{RESET}}$ pin.

POW — Power-Up Reset
Reset caused by the power-up reset circuit.

SW — Software Watchdog Reset
Reset caused by the software watchdog circuit.

HLT — Halt Monitor Reset
Reset caused by the halt monitor.

SYS — System Reset
The CPU16 does not support this function. This bit will never be set.

TST — Test Submodule Reset
Reset caused by the test submodule. Used during factory test reserved operating mode only.

D.2.5 SCIM Test Register E

SCIMTRE — Single-Chip Integration Module Test Register E **$YFFA08**

Used for factory test only.

D.2.6 Port A and B Data Registers

PORTA — Port A Data Register **$YFFA0A**
PORTB — Port B Data Register **$YFFA0B**

15 … 8	7	6	5	4	3	2	1	0
PA7 PA6 PA5 PA4 PA3 PA2 PA1 PA0	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
RESET:								
U U U U U U U U	U	U	U	U	U	U	U	U

Ports A and B are available in single-chip mode only. PORTA and PORTB can be read or written any time the MCU is not in emulator mode.

D.2.7 Port G and H Data Registers

PORTG — Port G Data Register **$YFFA0C**
PORTH — Port H Data Register **$YFFA0D**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PG7	PG6	PG5	PG4	PG3	PG2	PG1	PG0	PH7	PH6	PH5	PH4	PH3	PH2	PH1	PH0
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

Port G is available in single-chip mode only. These pins are always configured for use as general-purpose I/O in single-chip mode.

Port H is available in single-chip and 8-bit expanded modes only. The function of these pins is determined by the operating mode. There is no pin assignment register associated with this port.

These port data registers can be read or written any time the MCU is not in emulation mode. Reset has no effect.

D.2.8 Port G and H Data Direction Registers

DDRG — Port G Data Direction Register **$YFFA0E**
DDRH— Port H Data Direction Register **$YFFA0F**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DDG7	DDG6	DDG5	DDG4	DDG3	DDG2	DDG1	DDG0	DDH7	DDH6	DDH5	DDH4	DDH3	DDH2	DDH1	DDH0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The bits in this register control the direction of the port pin drivers when pins are configured as I/O. Setting a bit configures the corresponding pin as an output. Clearing a bit configures the corresponding pin as an input.

D.2.9 Port E Data Register

PORTE0 — Port E0 Data Register **$YFFA10**
PORTE1 — Port E1 Data Register **$YFFA12**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
RESET:															
								U	U	U	U	U	U	U	U

This register can be accessed in two locations and can be read or written at any time. A write to this register is stored in an internal data latch, and if any pin in the corresponding port is configured as an output, the value stored for that bit is driven out on the pin. A read of this data register returns the value at the pin only if the pin is configured as a discrete input. Otherwise, the value read is the value stored in the register. Bits [15:8] are unimplemented and will always read zero.

D.2.10 Port E Data Direction Register

DDRAB — Port A/B Data Direction Register **$YFFA14**
DDRE — Port E Data Direction Register **$YFFA15**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	DDA	DDB	DDE7	DDE6	DDE5	DDE4	DDE3	DDE2	DDE1	DDE0
RESET:								0	0	0	0	0	0	0	0

The port E data direction register controls the direction of the port E pin drivers when pins are configured for I/O. Setting a bit configures the corresponding pin as an output; clearing a bit configures the corresponding pin as an input. This register can be read or written at any time.

The port A/B data direction register controls the direction of the pin drivers for ports A and B, respectively, when the pins are configured for I/O. Setting DDA or DDB to one configures all pins in the corresponding port as outputs. Clearing DDA or DDB to zero configures all pins in the corresponding port as inputs. Bits [15:10] are unimplemented and will always read zero.

D.2.11 Port E Pin Assignment Register

PEPAR — Port E Pin Assignment **$YFFA17**

15	8	7	6	5	4	3	2	1	0
NOT USED		PEPA 7	PEPA 6	PEPA 5	PEPA 4	PEPA 3	PEPA 2	PEPA 1	PEPA 0
RESET:		DATA 8	DATA 8	DATA 8	DATA 8	DATA 8	DATA 8	DATA 8	DATA 8

This register determines the function of port E pins. Setting a bit assigns the corresponding pin to a bus control signal; clearing a bit assigns the pin to I/O port E. PE3 is not connected to a pin. PEPA3 can be read and written, but has no function. Bits [15:8] are unimplemented and will always read zero. **Table D-4** displays port E pin assignments.

Table D-4 Port E Pin Assignments

PEPAR Bit	Port E Signal	Bus Control Signal
PEPA7	PE7	SIZ1
PEPA6	PE6	SIZ0
PEPA5	PE5	$\overline{AS}$
PEPA4	PE4	$\overline{DS}$
PEPA3	PE3	$\overline{RMC}$[1]
PEPA2	PE2	$\overline{AVEC}$
PEPA1	PE1	$\overline{DSACK1}$
PEPA0	PE0	$\overline{DSACK0}$

NOTES:
1. $\overline{\text{RMC}}$ is a CPU32 control signal. The CPU16 does not use $\overline{\text{RMC}}$, thus $\overline{\text{RMC}}$/PE3 can be used for digital I/O at all times.

D.2.12 Port F Data Register

PORTF0— Port F Data Register 0 **$YFFA19**
PORTF1— Port F Data Register 1 **$YFFA1B**

15 … 8	7	6	5	4	3	2	1	0
NOT USED	PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0
RESET:	U	U	U	U	U	U	U	U

This register can be accessed in two locations and can be read or written at any time. A write to this register is stored in an internal data latch, and if any pin in the corresponding port is configured as an output, the value stored for that bit is driven out on the pin. A read of this data register returns the value at the pin only if the pin is configured as a discrete input. Otherwise, the value read is the value stored in the register. Bits [15:8] are unimplemented and will always read zero.

D.2.13 Port F Data Direction Register

DDRF — Port F Data Direction Register **$YFFA1D**

15 … 8	7	6	5	4	3	2	1	0
NOT USED	DDF7	DDF6	DDF5	DDF4	DDF3	DDF2	DDF1	DDF0
RESET:	0	0	0	0	0	0	0	0

This register controls the direction of the port F pin drivers when pins are configured for I/O. Setting a bit configures the corresponding pin as an output; clearing a bit configures the corresponding pin as an input. This register can be read or written at any time. Bits [15:8] are unimplemented and will always read zero.

D.2.14 Port F Pin Assignment Register

PFPAR — Port F Pin Assignment Register **$YFFA1F**

15 … 8	7	6	5	4	3	2	1	0
NOT USED	PFPA7	PFPA6	PFPA5	PFPA4	PFPA3	PFPA2	PFPA1	PFPA0
RESET:								
8- AND 16-BIT EXPANDED MODES	DATA9	DATA9	DATA9	DATA9	DATA9	DATA9	DATA9	DATA9
SINGLE-CHIP MODE	0	0	0	0	0	0	0	0

This register determines the function of port F pins. Setting a bit assigns the corresponding pin to a control signal; clearing a bit assigns the pin to port F. Bits [15:8] are

unimplemented and will always read zero. **Table D-5** shows port F pin assignments. **Table D-6** shows PFPAR pin functions.

Table D-5 Port F Pin Assignments

PFPAR Field	Port F Signal	Alternate Signal
PFPA3	PF[7:6]	$\overline{\text{IRQ[7:1]}}$
PFPA2	PF[5:4]	$\overline{\text{IRQ[5:4]}}$
PFPA1	PF[3:2]	$\overline{\text{IRQ[3:2]}}$
PFPA0	PF[1:0]	$\overline{\text{IRQ1}}$, FASTREF

Table D-6 PFPAR Pin Functions

PFPAx[1:0]	Port F Signal
00	I/O pin without edge detect
01	Rising edge detect
10	Falling edge detect
11	Interrupt request

D.2.15 System Protection Control Register

SYPCR — System Protection Control Register **$YFFA20**

15 … 8	7	6	5	4	3	2	1	0
NOT USED	SWE	SWP	SWT[1:0]		HME	BME	BMT[1:0]	
RESET:	1	$\overline{\text{MOD-CLK}}$	0	0	0	0	0	0

This register controls system monitor functions, software watchdog clock prescaling, and bus monitor timing. This register can be written once following power-on or reset. Bits [15:8] are unimplemented and will always read zero.

SWE — Software Watchdog Enable

0 = Software watchdog is disabled.
1 = Software watchdog is enabled.

SWP — Software Watchdog Prescaler

This bit controls the value of the software watchdog prescaler.

0 = Software watchdog clock is not prescaled.
1 = Software watchdog clock is prescaled by 512.

The reset value of SWP is the complement of the state of the MODCLK pin during reset.

SWT[1:0] — Software Watchdog Timing

This field selects the divide ratio used to establish the software watchdog timeout period. Refer to **Table D-7**.

Table D-7 Software Watchdog Divide Ratio

SWP	SWT[1:0]	Divide Ratio
0	00	2^9
0	01	2^{11}
0	10	2^{13}
0	11	2^{15}
1	00	2^{18}
1	01	2^{20}
1	10	2^{22}
1	11	2^{24}

The following equation calculates the timeout period for a slow reference frequency, where f_{ref} is equal to the EXTAL crystal frequency.

$$\text{Timeout Period} = \frac{\text{Divide Ratio Specified by SWP and SWT[1:0]}}{f_{ref}}$$

The following equation calculates the timeout period for a fast reference frequency.

$$\text{Timeout Period} = \frac{(128)(\text{Divide Ratio Specified by SWP and SWT[1:0]})}{f_{ref}}$$

The following equation calculates the timeout period for an externally input clock frequency on both slow and fast reference frequency devices, when f_{sys} is equal to the system clock frequency.

$$\text{Timeout Period} = \frac{\text{Divide Ratio Specified by SWP and SWT[1:0]}}{f_{sys}}$$

HME — Halt Monitor Enable
0 = Halt monitor is disabled.
1 = Halt monitor is enabled.

BME — Bus Monitor External Enable
0 = Disable bus monitor for external bus cycles.
1 = Enable bus monitor for external bus cycles.

BMT[1:0] — Bus Monitor Timing
This field selects the bus monitor timeout period. Refer to **Table D-8**.

Table D-8 Bus Monitor Period

BMT[1:0]	Bus Monitor Timeout Period
00	64 System clocks
01	32 System clocks
10	16 System clocks
11	8 System clocks

D.2.16 Periodic Interrupt Control Register

PICR — Periodic Interrupt Control Register **$YFFA22**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	PIRQL[2:0]			PIV[7:0]							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

PICR sets the interrupt level and vector number for the periodic interrupt timer (PIT). Bits [10:0] can be read or written at any time. Bits [15:11] are unimplemented and always read zero.

PIRQL[2:0] — Periodic Interrupt Request Level

This field determines the priority of periodic interrupt requests. A value of %000 disables PIT interrupts.

PIV[7:0] — Periodic Interrupt Vector

This field specifies the periodic interrupt vector number supplied by the SCIM2 when the CPU16 acknowledges an interrupt request.

D.2.17 Periodic Interrupt Timer Register

PITR — Periodic Interrupt Timer Register **$YFFA24**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	PTP	PITM[7:0]							
RESET:															
0	0	0	0	0	0	0	$\overline{\text{MOD-CLK}}$	0	0	0	0	0	0	0	0

Contains the count value for the periodic timer. This register can be read or written at any time.

PTP — Periodic Timer Prescaler

0 = Periodic timer clock not prescaled.

1 = Periodic timer clock prescaled by a value of 512.

PITM[7:0] — Periodic Interrupt Timing Modulus

This field determines the periodic interrupt rate. Use the following equations to calculate timer period.

The following equation calculates the PIT period when a slow reference frequency is used:

$$\text{PIT Period} = \frac{(\text{PITM}[7:0])(1 \text{ if PTP} = 0, 512 \text{ if PTP} = 1)(4)}{f_{ref}}$$

The following equation calculates the PIT period in fast reference mode:

$$\text{PIT Period} = \frac{(128)(\text{PITM}[7:0])(1 \text{ if PTP} = 0, 512 \text{ if PTP} = 1)(4)}{f_{ref}}$$

The following equation calculates the PIT period in external clock mode:

$$\text{PIT Period} = \frac{(\text{PITM}[7:0])(1 \text{ if PTP} = 0, 512 \text{ if PTP} = 1)(4)}{f_{ref}}$$

D.2.18 Software Watchdog Service Register

SWSR — Software Watchdog Service Register[1] **$YFFA26**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								SWSR[7:0]							
RESET:								0	0	0	0	0	0	0	0

NOTES:
1. This register is shown with a read value.

This register can be read or written at any time. Bits [15:8] are unimplemented and will always read zero.

To reset the software watchdog:

- Write $55 to SWSR.
- Write $AA to SWSR.

Both writes must occur in the order specified before the software watchdog times out, but any number of instructions can occur between the two writes.

D.2.19 Port F Edge-Detect Flag Register

PORTFE — Port F Edge-Detect Flag Register **$YFFA28**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								PEF7	EF6	RESERVED					PEF0
RESET:								0	0	0	0	0	0	0	0

When the corresponding pin is configured for edge detection, a PORTFE bit is set if an edge is detected. PORTFE bits remain set, regardless of the subsequent state of the corresponding pin, until cleared. To clear a bit, first read PORTFE, then write the bit to zero. When a pin is configured for general-purpose I/O or for use as an interrupt request input, PORTFE bits do not change state. Bits [15:8] are unimplemented and will always read zero.

D.2.20 Port F Edge-Detect Interrupt Vector

PFIVR — Port F Edge-Detect Interrupt Vector Register **$YFFA2A**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								PFIVR[7:0]							
RESET:								0	0	0	0	0	0	0	0

This register determines which vector in the exception vector table is used for interrupts generated by the port F edge-detect logic. Program PFIVR[7:0] to the value pointing to the appropriate interrupt vector. Bits [15:8] are unimplemented and will always read zero.

D.2.21 Port F Edge-Detect Interrupt Level

PFLVR — Port F Edge-Detect Interrupt Level Register **$YFFA2C**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								0	0	0	0	0	PFLV[2:0]		
RESET:								0	0	0	0	0	0	0	0

This register determines the priority level of the port F edge-detect interrupt. The reset value is $00, indicating that the interrupt is disabled. When several sources of interrupts from the SCIM are arbitrating for the same level, the port F edge-detect interrupt has the lowest arbitration priority. Bits [15:8] are unimplemented and will always read zero.

D.2.22 Port C Data Register

PORTC — Port C Data Register **$YFFA41**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								0	PC6	PC5	PC4	PC3	PC2	PC1	PC0
RESET:								0	1	1	1	1	1	1	1

PORTC latches data for chip-select pins configured as discrete outputs.

D.2.23 Chip-Select Pin Assignment Registers

CSPAR0 — Chip-Select Pin Assignment Register 0 **$YFFA44**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	CS5PA[1:0]		CS4PA[1:0]		CS3PA[1:0]		CS2PA[1:0]		CS1PA[1:0]		CS0PA[1:0]		CSBTPA[1:0]	
RESET:															
0	0	DATA 2	1	DATA 2	1	DATA 2	1	DATA 1	1	DATA 1	1	DATA 1	1	1	DATA 0

Chip-select pin assignment registers configure the chip-select pins for discrete I/O, an alternate function, or as an 8-bit or 16-bit chip-select. The possible encodings for each 2-bit field in CSPAR[0:1] (except for CSBTPA[1:0]) are shown in **Table D-9**.

Table D-9 Pin Assignment Field Encoding

CSxPA[1:0]	Description
00	Discrete output
01	Alternate function
10	Chip-select (8-bit port)
11	Chip-select (16-bit port)

This register contains seven 2-bit fields that determine the function of corresponding chip-select pins. Bits [15:14] are not used. These bits always read zero; writes have no effect. CSPAR0 bit 1 always reads one; writes to CSPAR0 bit 1 have no effect. The alternate functions can be enabled by data bus mode selection during reset. This register may be read or written at any time. After reset, software may enable one or more pins as discrete outputs.

Table D-10 shows CSPAR0 pin assignments.

Table D-10 CSPAR0 Pin Assignments

CSPAR0 Field	Chip-Select Signal	Alternate Signal	Discrete Output
CS5PA[1:0]	CS5	FC2	PC2
CS4PA[1:0]	CS4	FC1	PC1
CS3PA[1:0]	CS3	FC0	PC0
CS2PA[1:0]	CS2	$\overline{\text{BGACK}}$	—
CS1PA[1:0]	CS1	$\overline{\text{BG}}$	—
CS0PA[1:0]	CS0	$\overline{\text{BR}}$	—
CSBTPA[1:0]	$\overline{\text{CSBOOT}}$	—	—

CSPAR1 — Chip-Select Pin Assignment Register 1 **$YFFA46**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	CS10PA[1:0]		CS9PA[1:0]		CS8PA[1:0]		CS7PA[1:0]		CS6PA[1:0]	
RESET:															
0	0	0	0	0	0	DATA 7[1]	1	DATA [7:6][1]	1	DATA [7:5][1]	1	DATA [7:4][1]	1	DATA [7:3][1]	1

NOTES:
1. Refer to **Table D-12** for CSPAR1 reset state information.

CSPAR1 contains five 2-bit fields that determine the functions of corresponding chip-select pins. Bits [15:10] are not used. These bits always read zero; writes have no effect. **Table D-11** shows CSPAR1 pin assignments, including alternate functions that can be enabled by data bus mode selection during reset.

Table D-11 CSPAR1 Pin Assignments

CSPAR1 Field	Chip-Select Signal	Alternate Signal	Discrete Output
CS10PA[1:0]	$\overline{CS10}$	ADDR23[1]	ECLK
CS9PA[1:0]	$\overline{CS9}$	ADDR22[1]	PC6
CS8PA[1:0]	$\overline{CS8}$	ADDR21[1]	PC5
CS7PA[1:0]	$\overline{CS7}$	ADDR20[1]	PC4
CS6PA[1:0]	$\overline{CS6}$	ADDR19	PC3

NOTES:
1. On the CPU16, ADDR[23:20] follow the logic state of ADDR19 unless externally driven.

The reset state of DATA[7:3] determines whether pins controlled by CSPAR1 are initially configured as high-order address lines or chip-selects. **Table D-12** shows the correspondence between DATA[7:3] and the reset configuration of $\overline{CS}$[10:6]/ ADDR[23:19]. This register may be read or written at any time. After reset, software may enable one or more pins as discrete outputs.

Table D-12 Reset Pin Function of $\overline{CS}$[10:6]

Data Bus Pins at Reset					Chip-Select/Address Bus Pin Function				
DATA7	DATA6	DATA5	DATA4	DATA3	$\overline{CS10}$/ ADDR23	$\overline{CS9}$/ ADDR22	$\overline{CS8}$/ ADDR21	$\overline{CS7}$/ ADDR20	$\overline{CS6}$/ ADDR19
1	1	1	1	1	$\overline{CS10}$	$\overline{CS9}$	$\overline{CS8}$	$\overline{CS7}$	$\overline{CS6}$
1	1	1	1	0	$\overline{CS10}$	$\overline{CS9}$	$\overline{CS8}$	$\overline{CS7}$	ADDR19
1	1	1	0	X	$\overline{CS10}$	$\overline{CS9}$	$\overline{CS8}$	ADDR20	ADDR19
1	1	0	X	X	$\overline{CS10}$	$\overline{CS9}$	ADDR21	ADDR20	ADDR19
1	0	X	X	X	$\overline{CS10}$	ADDR22	ADDR21	ADDR20	ADDR19
0	X	X	X	X	ADDR23	ADDR22	ADDR21	ADDR20	ADDR19

D.2.24 Chip-Select Base Address Register Boot

CSBARBT — Chip-Select Base Address Register Boot **$YFFA48**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR 23	ADDR 22	ADDR 21	ADDR 20	ADDR 19	ADDR 18	ADDR 17	ADDR 16	ADDR 15	ADDR 14	ADDR 13	ADDR 12	ADDR 11	BLKSZ[2:0]		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

D.2.25 Chip-Select Base Address Registers

CSBAR[0:10] — Chip-Select Base Address Registers **$YFFA4C–$YFFA74**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR 23	ADDR 22	ADDR 21	ADDR 20	ADDR 19	ADDR 18	ADDR 17	ADDR 16	ADDR 15	ADDR 14	ADDR 13	ADDR 12	ADDR 11	BLKSZ[2:0]		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Each chip-select pin has an associated base address register. A base address is the lowest address in the block of addresses enabled by a chip select. CSBARBT contains the base address for selection of a boot memory device. Bit and field definitions for CSBARBT and CSBAR[0:10] are the same, but reset block sizes differ. These registers may be read or written at any time.

ADDR[23:11] — Base Address

This field sets the starting address of a particular chip-select's address space. The address compare logic uses only the most significant bits to match an address within a block. The value of the base address must be an integer multiple of the block size. Base address register diagrams show how base register bits correspond to address lines.

BLKSZ[2:0] — Block Size Field

This field determines the size of the block that is enabled by the chip-select.

Table D-13 shows bit encoding for the base address registers block size field.

Table D-13 Block Size Encoding

BLKSZ[2:0]	Block Size	Address Lines Compared
000	2 Kbytes	ADDR[23:11]
001	8 Kbytes	ADDR[23:13]
010	16 Kbytes	ADDR[23:14]
011	64 Kbytes	ADDR[23:16]
100	128 Kbytes	ADDR[23:17]
101	256 Kbytes	ADDR[23:18]
110	512 Kbytes	ADDR[23:19]
111	512 Kbytes[1]	ADDR[23:20]

NOTES:

1. On CPU16-based MCUs, ADDR[23:10] always follow the state of ADDR19, thus limiting the maximum usable block size to 512 Kbytes.

D.2.26 Chip-Select Option Register Boot

CSORBT — Chip-Select Option Register Boot **$YFFA4A**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE	BYTE[1:0]		R/$\overline{W}$[1:0]		STRB	$\overline{DSACK}$[3:0]				SPACE[1:0]		IPL[2:0]			$\overline{AVEC}$
RESET:															
0	1	1	1	1	0	1	1	0	1	1	1	0	0	0	0

D.2.27 Chip-Select Option Registers

CSOR[0:10] — Chip-Select Option Registers **$YFFA4E–YFFA76**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE	BYTE[1:0]		R/$\overline{W}$[1:0]		STRB	$\overline{DSACK}$[3:0]				SPACE[1:0]		IPL[2:0]			$\overline{AVEC}$
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CSORBT and CSOR[0:10] contain parameters that support operations from external memory devices. Bit and field definitions for CSORBT and CSOR[0:10] are the same.

MODE — Asynchronous/Synchronous Mode

0 = Asynchronous mode is selected.

1 = Synchronous mode is selected, and used with ECLK peripherals.

In asynchronous mode, chip-select assertion is synchronized with $\overline{AS}$ and $\overline{DS}$.

In synchronous mode, the chip-select signal is asserted with ECLK.

BYTE[1:0] — Upper/Lower Byte Option

This field is used only when the chip-select 16-bit port option is selected in the pin assignment register. This allows the usage of two external 8-bit memory devices to be concatenated to form a 16-bit memory. **Table D-14** shows upper/lower byte options.

Table D-14 BYTE Field Bit Encoding

BYTE[1:0]	Description
00	Disable
01	Lower byte
10	Upper byte
11	Both bytes

R/$\overline{W}$[1:0]— Read/Write

This field causes a chip-select to be asserted only for a read, only for a write, or for both reads and writes. **Table D-15** shows the options.

Table D-15 Read/Write Field Bit Encoding

R/$\overline{W}$[1:0]	Description
00	Disable
01	Read only
10	Write only
11	Read/Write

STRB — Address Strobe/Data Strobe

This bit controls the timing for assertion of a chip-select in asynchronous mode only. Selecting address strobe causes the chip-select to be asserted synchronized with address strobe. Selecting data strobe causes the chip-select to be asserted synchronized with data strobe. Data strobe timing is used to create a write strobe when needed.

0 = Address strobe
1 = Data strobe

$\overline{DSACK}$[3:0] — Data Strobe Acknowledge

This field specifies the source of $\overline{DSACK}$ in asynchronous mode as internally generated or externally supplied. It also allows the user to adjust bus timing with internal $\overline{DSACK}$ generation by controlling the number of wait states that are inserted to optimize bus speed in a particular application. **Table D-16** shows the $\overline{DSACK}$[3:0] field encoding.The fast termination encoding (%1110) effectively corresponds to –1 wait states.

Table D-16 $\overline{DSACK}$ Field Encoding

$\overline{DSACK}$[3:0]	Clock Cycles Required Per Access	Wait States Inserted Per Access
0000	3	0
0001	4	1
0010	5	2
0011	6	3
0100	7	4
0101	8	5
0110	9	6
0111	10	7
1000	11	8
1001	12	9
1010	13	10
1011	14	11
1100	15	12
1101	16	13
1110	2	–1 (Fast termination)
1111	—	External $\overline{DSACK}$

SPACE[1:0] — Address Space Select

Use this option field to select an address space for chip-select assertion or to configure

a chip-select as an interrupt acknowledge strobe for an external device. The CPU16 normally operates in supervisor mode only, but interrupt acknowledge cycles take place in CPU space.**Table D-17** shows address space bit encodings.

Table D-17 Address Space Bit Encodings

SPACE[1:0]	Address Space
00	CPU Space
01	User Space
10	Supervisor Space
11	Supervisor/User Space

IPL[2:0] — Interrupt Priority Level

When SPACE[1:0] is set for CPU space (%00), chip-select logic can be used as an interrupt acknowledge strobe for an external device. During an interrupt acknowledge cycle, the interrupt priority level is driven on address lines ADDR[3:1] is then compared to the value in IPL[2:0]. If the values match, an interrupt acknowledge strobe will be generated on the particular chip-select pin, provided other option register conditions are met. **Table D-18** shows IPL[2:0] field encoding.

Table D-18 Interrupt Priority Level Field Encoding

IPL[2:0]	Interrupt Priority Level
000	Any Level[1]
001	1
010	2
011	3
100	4
101	5
110	6
111	7

NOTES:
1. Any level means that chip-select is asserted regardless of the level of the interrupt acknowledge cycle.

$\overline{\text{AVEC}}$ — Autovector Enable

This field selects one of two methods of acquiring an interrupt vector during an interrupt acknowledge cycle. This field is not applicable when SPACE[1:0] = %00.

0 = External interrupt vector enabled
1 = Autovector enabled

If the chip select is configured to trigger on an interrupt acknowledge cycle (SPACE[1:0] = %00) and the $\overline{\text{AVEC}}$ field is set to one, the chip-select automatically generates $\overline{\text{AVEC}}$ and completes the interrupt acknowledge cycle. Otherwise, the vector must be supplied by the requesting external device to complete the IACK read cycle.

D.2.28 Master Shift Registers

TSTMSRA — Test Module Master Shift Register A **$YFFA30**

Used for factory test only.

TSTMSRB — Test Module Master Shift Register B **$YFFA32**

Used for factory test only.

D.2.29 Test Module Shift Count Register

TSTSC — Test Module Shift Count **$YFFA34**

Used for factory test only.

D.2.30 Test Module Repetition Count Register

TSTRC — Test Module Repetition Count **$YFFA36**

Used for factory test only.

D.2.31 Test Module Control Register

CREG — Test Module Control Register **$YFFA38**

Used for factory test only.

D.2.32 Test Module Distributed Register

DREG — Test Module Distributed Register **$YFFA3A**

Used for factory test only.

D.3 Standby RAM Module

Table D-19 shows the SRAM address map.

Table D-19 SRAM Address Map

Address[1]	15 0
$YFFB00	RAM Module Configuration Register (RAMMCR)
$YFFB02	RAM Test Register (RAMTST)
$YFFB04	RAM Array Base Address Register High (RAMBAH)
$YFFB06	RAM Array Base Address Register Low (RAMBAL)

NOTES:
1. Y = M111, where M is the logic state of the module mapping (MM) bit in the SCIMCR.

D.3.1 RAM Module Configuration Register

RAMMCR — RAM Module Configuration Register **$YFFB00**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STOP	0	0	PDS	RLCK	0	RASP[1:0]		NOT USED							
RESET:															
1	0	0	0	0	0	1	1								

STOP — Low-Power Stop Mode Enable

0 = SRAM operates normally.

1 = SRAM enters low-power stop mode.

This bit controls whether SRAM operates normally or enters low-power stop mode. In low-power stop mode, the array retains its contents, but cannot be read or written. This bit can be read or written at any time.

PDS — Power Down Status

0 = Voltage on V_{STBY} pin has fallen below valid standby level.

1 = Valid standby voltage on V_{STBY} pin.

Software can also write PDS to zero if it is necessary at some point to disable V_{STBY} monitoring.

RLCK — RAM Base Address Lock

0 = SRAM base address registers can be written.

1 = SRAM base address registers are locked and cannot be modified.

RLCK defaults to zero on reset; it can be written once to a one, and may be read at any time.

RASP[1:0] — RAM Array Space

The RASP field limits access to the SRAM array in microcontrollers that support separate user and supervisor operating modes. RASP1 has no effect because the CPU16 operates in supervisor mode only. This bit may be read or written at any time. Refer to **Table D-20**.

Table D-20 SRAM Array Address Space Type

RASP[1:0]	Space
X0	Program and data accesses
X1	Program access only

D.3.2 RAM Test Register

RAMTST — RAM Test Register **$YFFB02**

Used for factory test only.

D.3.3 Array Base Address Registers

RAMBAH — Array Base Address Register High **$YFFB04**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								ADDR 23	ADDR 22	ADDR 21	ADDR 20	ADDR 19	ADDR 18	ADDR 17	ADDR 16
RESET:								0	0	0	0	0	0	0	0

RAMBAL — Array Base Address Register Low (2K — R1/916R1) **$YFFB06**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR 15	ADDR 14	ADDR 13	ADDR 12	ADDR 11	0	0	0	0	0	0	0	0	0	0	0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RAMBAL — Array Base Address Register Low (4K — R3/916R3) **$YFFB06**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR 15	ADDR 14	ADDR 13	ADDR 12	0	0	0	0	0	0	0	0	0	0	0	0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RAMBAH and RAMBAL specify the SRAM array base address in the system memory map. They can only be written while the SRAM is in low-power stop mode (STOP = 1, the default out of reset) and the base address lock is disabled (RLCK = 0, the default out of reset). This prevents accidental remapping of the array. Because the CPU16 drives ADDR[23:20] to the same logic level as ADDR19, the values of the RAMBAH ADDR[23:20] fields must match the value of the ADDR19 field for the array to be accessible. These registers may be read at any time. RAMBAH[15:8] are unimplemented and will always read zero.

D.4 Masked ROM Module

Table D-21 shows the MRM address map. The MC68HC16R1 has a 48-Kbyte MRM. The MC68HC16R3 has one 32-Kbyte MRM and one 64-Kbyte MRM.

The reset states shown for the MRM registers are for the generic (blank ROM) versions of the device. Several MRM register bit fields can be user-specified on a custom masked ROM device. Contact a Motorola sales representative for information on ordering a custom ROM device.

Table D-21 MRM Address Map

Address[1]	15 0	Module
$YFF820	Masked ROM Module Configuration Register (MR1MCR)	Masked ROM1 (R1 — 48K) (R3 — 32K)
$YFF822	Not Implemented	
$YFF824	ROM Array Base Address Register High (ROM1BAH)	
$YFF826	ROM Array Base Address Register Low (ROM1BAL)	
$YFF828	Signature Register High (R1SIGHI)	
$YFF82A	Signature Register Low (R1SIGLO)	
$YFF82C	Not Implemented	
$YFF82E	Not Implemented	
$YFF830	ROM Bootstrap Word 0 (ROM1BS0)	
$YFF832	ROM Bootstrap Word 1 (ROM1BS1)	
$YFF834	ROM Bootstrap Word 2 (ROM1BS2)	
$YFF836	ROM Bootstrap Word 3 (ROM1BS3)	
$YFF838	Not Implemented	
$YFF83A	Not Implemented	
$YFF83C	Not Implemented	
$YFF83E	Not Implemented	
$YFF840	Masked ROM Module Configuration Register (MR2MCR)	Masked ROM2 (R3 — 64K)
$YFF842	Not Implemented	
$YFF844	ROM Array Base Address Register High (ROM2BAH)	
$YFF846	ROM Array Base Address Register Low (ROM2BAL)	
$YFF848	Signature Register High (R2SIGHI)	
$YFF84A	Signature Register Low (R2SIGLO)	
$YFF84C	Not Implemented	
$YFF84E	Not Implemented	
$YFF850	ROM Bootstrap Word 0 (ROM2BS0)	
$YFF852	ROM Bootstrap Word 1 (ROM2BS1)	
$YFF854	ROM Bootstrap Word 2 (ROM2BS2)	
$YFF856	ROM Bootstrap Word 3 (ROM2BS3)	
$YFF858	Not Implemented	
$YFF85A	Not Implemented	
$YFF85C	Not Implemented	
$YFF85E	Not Implemented	

NOTES:
1. Y = M111, where M is the logic state of the module mapping (MM) bit in the SCIMCR.

NOTE

The following register descriptions apply to the corresponding register in any of the control blocks. MRxMCR, for example, refers to MR1MCR on both the MC68HC16R1 and the MC68HC16R3 and MR2MCR on the MC68HC16R3.

D.4.1 Masked ROM Module Configuration Register

MRxMCR — Masked ROM Module Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STOP	0	0	$\overline{\text{BOOT}}$	LOCK	EMUL	ASPC[1:0]		WAIT[1:0]		NOT USED					
RESET:															
$\overline{\text{DATA14}}$	0	0	1	0	0	1	1	1	1						

Table D-22 shows MRxMCR address locations.

Table D-22 MRxMCR Registers

MC68HC16R1	MC68HC16 R3	Address
MR1MCR	MR1MCR	$YFF820
—	MR2MCR	$YFF840

STOP — Low-Power Stop Mode Enable

The reset state of the STOP bit is the complement of DATA14 state during reset. The ROM array base address cannot be changed unless the STOP bit is set.

0 = ROM array operates normally.

1 = ROM array operates in low-power stop mode. The ROM array cannot be read in this mode.

This bit may be read or written at any time.

$\overline{\text{BOOT}}$— Boot ROM Control

Reset state of $\overline{\text{BOOT}}$ is specified at mask time. This is a read-only bit.

0 = ROM responds to bootstrap word locations during reset vector fetch.

1 = ROM does not respond to bootstrap word locations during reset vector fetch.

Bootstrap operation is overridden if STOP = 1 at reset.

LOCK — Lock Registers

The reset state of LOCK is specified at mask time. If the reset state of the LOCK is zero, it can be set once after reset to allow protection of the registers after initialization. Once the LOCK bit is set, it cannot be cleared again until after a reset. LOCK protects the ASPC and WAIT fields, as well as the ROMBAL and ROMBAH registers. ASPC, ROMBAL and ROMBAH are also protected by the STOP bit.

0 = Write lock disabled. Protected registers and fields can be written.

1 = Write lock enabled. Protected registers and fields cannot be written.

EMUL — Emulation Mode Control

0 = Normal ROM operation

1 = Accesses to the ROM array are forced external, allowing memory selected by the CSM pin to respond to the access.

Since the MC68HC16R1 does not support ROM emulation mode, this bit should never be set.

ASPC[1:0] — ROM Array Space

The ASPC field limits access to the SRAM array in microcontrollers that support separate user and supervisor operating modes. ASPC1 has no effect because the CPU16 operates in supervisor mode only. This bit may be read or written at any time. The reset state of ASPC[1:0] is specified at mask time. **Table D-23** shows ASPC[1:0] encoding.

Table D-23 ROM Array Space Field

ASPC[1:0]	State Specified
X0	Program and data accesses
X1	Program access only

WAIT[1:0] — Wait States Field

WAIT[1:0] specifies the number of wait states inserted by the MRM during ROM array accesses. The reset state of WAIT[1:0] is user specified. The field can be written only if LOCK = 0 and STOP = 1. **Table D-24** shows the wait states field.

Table D-24 Wait States Field

WAIT[1:0]	Number of Wait States	Clocks per Bus Cycle
00	0	3
01	1	4
10	2	5
11	–1	2

D.4.2 ROM Array Base Address Registers

ROMxBAH — ROM Array Base Address Register High[1]

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	ADDR 23	ADDR 22	ADDR 21	ADDR 20	ADDR 19	ADDR 18	ADDR 17	ADDR 16
RESET:								1	1	1	1	1	1	1	1

NOTES:

1. Reset value of the shaded bits is user specified but the bits can be written after reset to change the base address. If the values of ROMxBAH bits ADDR[23:20] do not match that of ADDR19, however, the CPU16 cannot access the ROM array.

ROM2BAL — ROM Array Base Address Register Low (64K —R3)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RESET:															
0	0	0													

ROM1BAL — ROM Array Base Address Register Low (48K — R1)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RESET:															
0	0	0													

ROM1BAL — ROM Array Base Address Register Low (32K — R3)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR 15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RESET:															
0	0	0													

Table D-25 shows ROMxBAH and ROMxBAL address locations.

Table D-25 ROMxBAH and ROMxBAL Registers

MC68HC16R1	MC68HC16 R3	Address
ROM1BAH	ROM1BAH	$YFF824
—	ROM2BAH	$YFF844
ROM1BAL	ROM1BAL	$YFF826
—	ROM2BAL	$YFF846

ROMxBAH and ROMxBAL specify ROM array base address. The reset state of these registers is specified at mask time. They can only be written when STOP = 1 and LOCK = 0. This prevents accidental remapping of the array.

Masked ROM arrays sized to an integral power of two must be mapped to a base address that is an integer multiple of the array size. Thus, a 32-Kbyte array must be mapped to a base address that is an integer multiple of $8000, and a 64-Kbyte array must be mapped to a base address that is an integer multiple of $10000.

The 48-Kbyte MRM, like the 64-Kbyte MRM, must be mapped to a base address that is an integer multiple of $10000. The array only occupies the first 48-Kbytes of the 64-Kbyte block of address space into which it is mapped. Other memory resources (subject to block size and base address constraints) can be mapped immediately following the 48-Kbyte array and will remain accessible.

D.4.3 ROM Signature Registers

RxSIGHI — ROM Signature High Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED													RSP18	RSP17	RSP16
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RxSIGLO — ROM Signature Low Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSP15	RSP14	RSP13	RSP12	RSP11	RSP10	RSP9	RSP8	RSP7	RSP6	RSP5	RSP4	RSP3	RSP2	RSP1	RSP0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table D-26 shows RxSIGHI and RxSIGLO address locations.

Table D-26 RxSIGHI and RxSIGLO Registers

MC68HC16R1	MC68HC16 R3	Address
R1SIGHI	R1SIGHI	$YFF828
—	R2SIGHI	$YFF848
R1SIGLO	R1SIGLO	$YFF82A
—	R2SIGLO	$YFF84A

RxSIGHI and RxSIGLO specify a ROM signature pattern. A user-written signature identification algorithm allows identification of the ROM array content. The signature is specified at mask time and cannot be changed.

D.4.4 ROM Bootstrap Words

ROMxBS0 — ROM Bootstrap Word 0 **$YFF830**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED				ZK[3:0]				SK[3:0]				PK[3:0]			

ROMxBS1 — ROM Bootstrap Word 1 **$YFF832**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC[15:0]															

ROMxBS2 — ROM Bootstrap Word 2 **$YFF834**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SP[15:0]															

ROMxBS3 — ROM Bootstrap Word 3 **$YFF836**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ROMxBS3 — ROM Bootstrap Word 3 **$YFF836**

IZ[15:0]

Table D-27 shows ROMxBS[0:3] address locations.

Table D-27 ROMxBS[0:3] Registers

MC68HC16R1	MC68HC16 R3	Address
ROM1BS0	ROM1BS0	$YFF830
—	ROM2BS0	$YFF850
ROM1BS1	ROM1BS1	$YFF832
—	ROM2BS1	$YFF852
ROM1BS2	ROM1BS2	$YFF834
—	ROM2BS2	$YFF854
ROM1BS3	ROM1BS3	$YFF836
—	ROM2BS3	$YFF856

Typically, CPU16 reset vectors reside in non-volatile memory and are only fetched when the CPU16 comes out of reset. These four words can be used as reset vectors with the contents specified at mask time. The content of these words cannot be changed. On generic (blank ROM) M68HC16 R-series devices, ROMxBS[0:3] are masked to $0000. When the ROM on M68HC16 R-series devices is masked with customer specific code, ROMxBS[0:3] respond to system addresses $00000 to $00006 during the reset vector fetch if $\overline{\text{BOOT}}$ = 0.

D.5 Analog-to-Digital Converter Module

Table D-28 shows the ADC module address map.

Table D-28 ADC Module Address Map

Address[1]	15 8	7 0
$YFF700	ADC Module Configuration Register (ADCMCR)	
$YFF702	ADC Test Register (ADCTEST)	
$YFF704	Not Used	
$YFF706	Not Used	Port ADA Data Register (PORTADA)
$YFF708	Not Used	
$YFF70A	Control Register 0 (ADCTL0)	
$YFF70C	Control Register 1 (ADCTL1)	
$YFF70E	Status Register (ADCSTAT)	
$YFF710	Right-Justified Unsigned Result Register 0 (RJURR0)	
$YFF712	Right-Justified Unsigned Result Register 1 (RJURR1)	
$YFF714	Right-Justified Unsigned Result Register 2 (RJURR2)	
$YFF716	Right-Justified Unsigned Result Register 3 (RJURR3)	
$YFF718	Right-Justified Unsigned Result Register 4 (RJURR4)	
$YFF71A	Right-Justified Unsigned Result Register 5 (RJURR5)	
$YFF71C	Right-Justified Unsigned Result Register 6 (RJURR6)	
$YFF71E	Right-Justified Unsigned Result Register 7 (RJURR7)	
$YFF720	Left-Justified Signed Result Register 0 (LJSRR0)	
$YFF722	Left-Justified Signed Result Register 1 (LJSRR1)	
$YFF724	Left-Justified Signed Result Register 2 (LJSRR2)	
$YFF726	Left-Justified Signed Result Register 3 (LJSRR3)	
$YFF728	Left-Justified Signed Result Register 4 (LJSRR4)	
$YFF72A	Left-Justified Signed Result Register 5 (LJSRR5)	
$YFF72C	Left-Justified Signed Result Register 6 (LJSRR6)	
$YFF72E	Left-Justified Signed Result Register 7 (LJSRR7)	
$YFF730	Left-Justified Unsigned Result Register 0 (LJURR0)	
$YFF732	Left-Justified Unsigned Result Register 1 (LJURR1)	
$YFF734	Left-Justified Unsigned Result Register 2 (LJURR2)	
$YFF736	Left-Justified Unsigned Result Register 3 (LJURR3)	
$YFF738	Left-Justified Unsigned Result Register 4 (LJURR4)	
$YFF73A	Left-Justified Unsigned Result Register 5 (LJURR5)	
$YFF73C	Left-Justified Unsigned Result Register 6 (LJURR6)	
$YFF73E	Left-Justified Unsigned Result Register 7 (LJURR7)	

NOTES:

1. Y = M111, where M is the logic state of the module mapping (MM) bit in the SCIMCR.

D.5.1 ADC Module Configuration Register

ADCMCR — ADC Module Configuration Register **$YFF700**

15	14	13	12 … 8	7	6 … 0
STOP	FRZ		NOT USED	SUPV	NOT USED
RESET:					
1	0	0		1	

ADCMCR controls ADC operation during low-power stop mode, background debug mode, and freeze mode.

STOP — Low-Power Stop Mode Enable

0 = Normal operation

1 = Low-power operation

STOP places the ADC in low-power state. Setting STOP aborts any conversion in progress. STOP is set to logic level one during reset, and may be cleared to logic level zero by the CPU16. Clearing STOP enables normal ADC operation. However, because analog circuitry bias current has been turned off, there is a period of recovery before output stabilization.

FRZ[1:0] — Freeze Assertion Response

The FRZ field determines ADC response to assertion of the FREEZE signal when the device is placed in background debug mode. Refer to **Table D-29**.

Table D-29 Freeze Encoding

FRZ[1:0]	Response
00	Ignore FREEZE, continue conversions
01	Reserved
10	Finish conversion in process, then freeze
11	Freeze immediately

SUPV — Supervisor/Unrestricted

This bit has no effect because the CPU16 always operates in supervisor mode.

D.5.2 ADC Test Register

ADCTEST — ADC Test Register **$YFF702**

Used for factory test only.

D.5.3 Port ADA Data Register

PORTADA — Port ADA Data Register **$YFF706**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								PADA 7	PADA 6	PADA 5	PADA 4	PADA 3	PADA 2	PADA 1	PADA 0
RESET:															
								REFLECTS STATE OF THE INPUT PINS							

Port ADA is an input port that shares pins with the A/D converter inputs.

PADA[5:0] — Port ADA Data Pins

A read of PADA[7:0] returns the logic level of the port ADA pins. If an input is not at an appropriate logic level (that is, outside the defined levels), the read is indeterminate. Use of a port ADA pin for digital input does not preclude its simultaneous use as an analog input.

D.5.4 Control Register 0

ADCTL0 — Control Register 0 **$YFF70A**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								RES10	STS[1:0]		PRS[4:0]				
RESET:								0	0	0	0	0	0	1	1

ADCTL0 is used to select 8- or 10-bit conversions, sample time, and ADC clock frequency. Writes to it have immediate effect.

RES10 — 10-Bit Resolution

0 = 8-bit conversion

1 = 10-bit conversion

Conversion results are appropriately aligned in result registers to reflect the number of bits.

STS[1:0] — Sample Time Selection

Total conversion time is the sum of initial sample time, transfer time, final sample time, and resolution time. Initial sample time is fixed at two ADC clocks. Transfer time is fixed at two ADC clocks. Resolution time is fixed at 10 ADC clocks for an 8-bit conversion and 12 ADC clocks for a 10-bit conversion. Final sample time is determined by the STS[1:0] field. Refer to **Table D-30**.

Table D-30 Sample Time Selection

STS[1:0]	Sample Time
00	2 ADC Clock Periods
01	4 ADC Clock Periods
10	8 ADC Clock Periods
11	16 ADC Clock Periods

PRS[4:0] — Prescaler Rate Selection

The ADC clock is derived from the system clock by a programmable prescaler. ADC clock period is determined by the value of the PRS field in ADCTL0. The prescaler has two stages. The first stage is a 5-bit modulus counter. It divides the system clock by any value from 2 to 32 (PRS[4:0] = %00000 to %11111). The second stage is a divide-by-two circuit. Refer to **Table D-31**.

Table D-31 Prescaler Output

PRS[4:0]	ADC Clock	Minimum System Clock	Maximum System Clock
%00000	Reserved	—	—
%00001	System Clock/4	2.0 MHz	8.4 MHz
%00010	System Clock/6	3.0 MHz	12.6 MHz
%00011	System Clock/8	4.0 MHz	16.8 MHz
…	…	…	…
%11101	System Clock/60	30.0 MHz	—
%11110	System Clock/62	31.0 MHz	—
%11111	System Clock/64	32.0 MHz	—

D.5.5 Control Register 1

ADCTL1 — Control Register 1 **$YFF70C**

15 … 7	6	5	4	3	2	1	0
NOT USED	SCAN	MULT	S8CM	CD	CC	CB	CA
RESET:	0	0	0	0	0	0	0

ADCTL1 is used to initiate an A/D conversion and to select conversion modes and a conversion channel or channels. It can be read or written at any time. A write to ADCTL1 initiates a conversion sequence. If a conversion sequence is already in progress, a write to ADCTL1 aborts it and resets the SCF and CCF flags in the ADC status register.

SCAN — Scan Mode Selection
0 = Single conversion
1 = Continuous conversions
Length of conversion sequence(s) is determined by S8CM.

MULT — Multichannel Conversion
0 = Conversion sequence(s) run on a single channel selected by [CD:CA].
1 = Sequential conversions of four or eight channels selected by [CD:CA].
Length of conversion sequence(s) is determined by S8CM.

S8CM — Select Eight-Conversion Sequence Mode
0 = Four-conversion sequence
1 = Eight-conversion sequence

This bit determines the number of conversions in a conversion sequence. **Table D-32** displays the different ADC conversion modes.

Table D-32 ADC Conversion Mode

SCAN	MULT	S8CM	MODE
0	0	0	Single 4-Conversion Single-Channel Sequence
0	0	1	Single 8-Conversion Single-Channel Sequence
0	1	0	Single 4-Conversion Multichannel Sequence
0	1	1	Single 8-Conversion Multichannel Sequence
1	0	0	Multiple 4-Conversion Single-Channel Sequences
1	0	1	Multiple 8-Conversion Single-Channel Sequences
1	1	0	Multiple 4-Conversion Multichannel Sequences
1	1	1	Multiple 8-Conversion Multichannel Sequences

CD:CA — Channel Selection

Bits in this field select input channel or channels for A/D conversion.

Conversion mode determines which channel or channels are selected for conversion and which result registers are used to store conversion results. **Tables D-33** and **D-34** contain a summary of the effects of ADCTL1 bits and fields.

Table D-33 Single-Channel Conversions (MULT = 0)

S8CM	CD	CC	CB	CA	Input	Result Register[1]
0	0	0	0	0	AN0	RSLT[0:3]
0	0	0	0	1	AN1	RSLT[0:3]
0	0	0	1	0	AN2	RSLT[0:3]
0	0	0	1	1	AN3	RSLT[0:3]
0	0	1	0	0	AN4	RSLT[0:3]
0	0	1	0	1	AN5	RSLT[0:3]
0	0	1	1	0	AN6	RSLT[0:3]
0	0	1	1	1	AN7	RSLT[0:3]
0	1	0	0	0	Reserved	RSLT[0:3]
0	1	0	0	1	Reserved	RSLT[0:3]
0	1	0	1	0	Reserved	RSLT[0:3]
0	1	0	1	1	Reserved	RSLT[0:3]
0	1	1	0	0	V_{RH}	RSLT[0:3]
0	1	1	0	1	V_{RL}	RSLT[0:3]
0	1	1	1	0	$(V_{RH} - V_{RL}) / 2$	RSLT[0:3]
0	1	1	1	1	Test/Reserved	RSLT[0:3]
1	0	0	0	0	AN0	RSLT[0:7]
1	0	0	0	1	AN1	RSLT[0:7]
1	0	0	1	0	AN2	RSLT[0:7]
1	0	0	1	1	AN3	RSLT[0:7]
1	0	1	0	0	AN4	RSLT[0:7]
1	0	1	0	1	AN5	RSLT[0:7]
1	0	1	1	0	AN6	RSLT[0:7]
1	0	1	1	1	AN7	RSLT[0:7]
1	1	0	0	0	Reserved	RSLT[0:7]
1	1	0	0	1	Reserved	RSLT[0:7]
1	1	0	1	0	Reserved	RSLT[0:7]
1	1	0	1	1	Reserved	RSLT[0:7]
1	1	1	0	0	V_{RH}	RSLT[0:7]
1	1	1	0	1	V_{RL}	RSLT[0:7]
1	1	1	1	0	$(V_{RH} - V_{RL}) / 2$	RSLT[0:7]
1	1	1	1	1	Test/Reserved	RSLT[0:7]

NOTES:

1. Result register (RSLT) is either RJURRX, LJSRRX, or LJURRX, depending on the address read.

Table D-34 Multiple-Channel Conversions (MULT = 1)

S8CM	CD	CC	CB	CA	Input	Result Register[1]
0	0	0	X	X	AN0	RSLT0
					AN1	RSLT1
					AN2	RSLT2
					AN3	RSLT3
0	0	1	X	X	AN4	RSLT0
					AN5	RSLT1
					AN6	RSLT2
					AN7	RSLT3
0	1	0	X	X	Reserved	RSLT0
					Reserved	RSLT1
					Reserved	RSLT2
					Reserved	RSLT3
0	1	1	X	X	V_{RH}	RSLT0
					V_{RL}	RSLT1
					$(V_{RH} - V_{RL}) / 2$	RSLT2
					Test/Reserved	RSLT3
1	0	X	X	X	AN0	RSLT0
					AN1	RSLT1
					AN2	RSLT2
					AN3	RSLT3
					AN4	RSLT4
					AN5	RSLT5
					AN6	RSLT6
					AN7	RSLT7
1	1	X	X	X	Reserved	RSLT0
					Reserved	RSLT1
					Reserved	RSLT2
					Reserved	RSLT3
					V_{RH}	RSLT4
					V_{RL}	RSLT5
					$(V_{RH} - V_{RL}) / 2$	RSLT6
					Test/Reserved	RSLT7

NOTES:
1. Result register (RSLT) is either RJURRX, LJSRRX, or LJURRX, depending on the address read.

D.5.6 Status Register

ADCSTAT — Status Register **$YFF70E**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCF	NOT USED				CCTR[2:0]			CCF[7:0]							
RESET:															
0					0	0	0	0	0	0	0	0	0	0	0

ADCSTAT contains information related to the status of a conversion sequence.

SCF — Sequence Complete Flag

0 = Sequence not complete

1 = Sequence complete

SCF is set at the end of the conversion sequence when SCAN is cleared, and at the end of the first conversion sequence when SCAN is set. SCF is cleared when ADCTL1 is written and a new conversion sequence begins.

CCTR[2:0] — Conversion Counter

This field reflects the contents of the conversion counter pointer in either four or eight count conversion sequence. The value corresponds to the number of the next result register to be written, and thus indicates which channel is being converted.

CCF[7:0] — Conversion Complete Flags

Each bit in this field corresponds to an A/D result register (for example, CCF7 to RSLT7). A bit is set when conversion for the corresponding channel is complete, and remains set until the associated result register is read.

D.5.7 Right Justified, Unsigned Result Register

RJURR — Right-Justified, Unsigned Result Register **$YFF710–$YFF71F**

15	10	9	8	7	6	5	4	3	2	1	0
NOT USED		10	10	8/10	8/10	8/10	8/10	8/10	8/10	8/10	8/10

Conversion result is unsigned right-justified data. Bits [9:0] are used for 10-bit resolution. For 8-bit conversions, bits [7:0] contain data and bits [9:8] are zero. Bits [15:10] always return zero when read.

D.5.8 Left Justified, Signed Result Register

LJSRR — Left Justified, Signed Result Register **$YFF720–$YFF72F**

15	14	13	12	11	10	9	8	7	6	5	0
8/10	8/10	8/10	8/10	8/10	8/10	8/10	8/10	10	10	NOT USED	

Conversion result is signed left-justified data. Bits [15:6] are used for 10-bit resolution. For 8-bit conversions, bits [15:8] contain data and bits [7:6] are zero. Although the ADC is unipolar, it is assumed that the zero point is halfway between low and high reference when this format is used ($V_{RH} – V_{RL}/2$). For positive input, bit 15 = 0. For negative input, bit 15 = 1. Bits [5:0] always return zero when read.

D.5.9 Left Justified, Unsigned Result Register

LJURR — Left Justified, Unsigned Result Register **$YFF730–$YFF73F**

15	14	13	12	11	10	9	8	7	6	5	0
8/10	8/10	8/10	8/10	8/10	8/10	8/10	8/10	10	10	NOT USED	

Conversion result is unsigned left-justified data. Bits [15:6] are used for 10-bit resolution. For 8-bit conversions, bits [15:8] contain data and bits [7:6] are zero. Bits [5:0] always return zero when read.

D.6 Multichannel Communication Interface Module

Table D-35 shows the MCCI address map.

Table D-35 MCCI Address Map

Address[1]	15 8	7 0
$YFFC00	MCCI Module Configuration Register (MMCR)	
$YFFC02	MCCI Test Register (MTEST)	
$YFFC04	SCI Interrupt Level Register (ILSCI)	MCCI Interrupt Vector Register (MIVR)
$YFFC06	SPI Interrupt Level Register (ILSPI)	Not Used
$YFFC08	Not Used	MCCI Pin Assignment Register (MPAR)
$YFFC0A	Not Used	MCCI Data Direction Register (MDDR)
$YFFC0C	Not Used	MCCI Port Data Register (PORTMC)
$YFFC0E	Not Used	MCCI Port Pin State Register (PORTMCP)
$YFFC10 – $YFFC16	Not Used	
$YFFC18	SCIA Control Register 0 (SCCR0A)	
$YFFC1A	SCIA Control Register 1 (SCCR1A)	
$YFFC1C	SCIA Status Register (SCSRA)	
$YFFC1E	SCIA Data Register (SCDRA)	
$YFFC20 – $YFFC26	Not Used	
$YFFC28	SCIB Control Register 0 (SCCR0B)	
$YFFC2A	SCIB Control Register 1 (SCCR1B)	
$YFFC2C	SCIB Status Register (SCSRB)	
$YFFC2E	SCIB Data Register (SCDRB)	
$YFFC30 – $YFFC36	Not Used	
$YFFC38	SPI Control Register (SPCR)	
$YFFC3A	Not Used	
$YFFC3C	SPI Status Register (SPSR)	
$YFFC3E	SPI Data Register (SPDR)	

NOTES:
1. Y = M111, where M is the logic state of the module mapping (MM) bit in the SCIMCR.

D.6.1 MCCI Module Configuration Register

MMCR — MCCI Module Configuration Register **$YFFC00**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STOP	NOT USED							SUPV	NOT USED			IARB[3:0]			
RESET:															
0								1				0	0	0	0

MMCR bits enable stop mode, establish the privilege level required to access certain MCCI registers, and determine the arbitration priority of MCCI interrupt requests.

STOP — Low-Power Stop Mode Enable

0 = MCCI clock operates normally.

1 = MCCI clock is stopped.

When STOP is set, the MCCI enters low-power stop mode. The system clock input to the module is disabled. While STOP is set, only MMCR reads and writes are guaranteed to be valid. Only writes to other MCCI registers are guaranteed valid. The SCI receiver and transmitter must be disabled before STOP is set. To stop the SPI, set the HALT bit in SPCR3, wait until the HALTA flag is set, then set STOP.

Bits [14:8] — Not Implemented

SUPV — Supervisor/Unrestricted

This bit has no effect because the CPU16 in the MCU operates only in supervisor mode.

Bits [6:4] — Not Implemented

IARB[3:0] — Interrupt Arbitration ID

The IARB field is used to arbitrate between simultaneous interrupt requests of the same priority. Each module that can generate interrupt requests must be assigned a unique, non-zero IARB field value.

D.6.2 MCCI Test Register

MTEST — MCCI Test Register **$YFFC02**

Used for factory test only.

D.6.3 SCI Interrupt Level Register

ILSCI — SCI Interrupt Level Register **$YFFC04**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	ILSCIB[2:0]			ILSCIA[2:0]			MIVR							
RESET:															
0	0	0	0	0	0	0	0								

Bits [15:14] — Not Implemented

ILSCIA[2:0], ILSCIB[2:0]— Interrupt Level for SCIA, SCIB

The values of ILSCIA[2:0] and ILSCIB[2:0] in ILSCI determine the interrupt request levels of SCIA and SCIB interrupts, respectively. Program this field to a value from $0 (interrupts disabled) through $7 (highest priority).

D.6.4 MCCI Interrupt Vector Register

MIVR — MCCI Interrupt Vector Register **$YFFC05**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ILSCI								INTV[7:2]						INTV[1:0]	
RESET:															
								0	0	0	0	1	1	1	1

The MIVR determines which three vectors in the exception vector table are to be used for MCCI interrupts. The SPI and both SCI interfaces have separate interrupt vectors adjacent to one another. When initializing the MCCI, program INTV[7:2] so that INTV[7:0] correspond to three of the user-defined vectors ($40–$FF). INTV[1:0] are determined by the serial interface causing the interrupt, and are set by the MCCI.

At reset, MIVR is initialized to $0F, which corresponds to the uninitialized interrupt vector in the exception table.

INTV[7:2] — Interrupt Vector

INTV[7:2] are the six high-order bits of the three MCCI interrupt vectors for the MCCI, as programmed by the user.

INTV[1:0] — Interrupt Vector Source

INTV[1:0] are the two low-order bits of the three interrupt vectors for the MCCI. They are automatically set by the MCCI to indicate the source of the interrupt. Refer to **Table D-36**.

Table D-36 Interrupt Vector Sources

INTV[1:0]	Source of Interrupt
00	SCIA
01	SCIB
10	SPI

Writes to INTV0 and INTV1 have no meaning or effect. Reads of INTV0 and INTV1 return a value of one.

D.6.5 SPI Interrupt Level Register

ILSPI — SPI Interrupt Level Register **$YFFC06**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	ILSPI[2:0]			0	0	0	NOT USED							
RESET:															
0	0	0	0	0	0	0	0								

The ILSPI determines the priority level of interrupts requested by the SPI.

Bits [15:14] — Not Implemented

ILSPI[2:0]— Interrupt Level for SPI

ILSPI[2:0] determine the interrupt request levels of SPI interrupts. Program this field to a value from $0 (interrupts disabled) through $7 (highest priority). If the interrupt-request level programmed in this field matches the interrupt-request level programmed for one of the SCI interfaces and both request an interrupt simultaneously, the SPI is given priority.

Bits [10:8] — Not Implemented

D.6.6 MCCI Pin Assignment Register

MPAR — MCCI Pin Assignment Register **$YFFC08**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED												MPA3	NOT USED	MPA1	MPA0
RESET:															
0	0	0	0	0	0	0	0					0		0	0

The MPAR determines which of the SPI pins, with the exception of the SCK pin, are actually used by the SPI submodule, and which pins are available for general-purpose I/O. The state of SCK is determined by the SPI enable bit in SPCR1. Clearing a bit in MPAR assigns the corresponding pin to general purpose I/O; setting a bit assigns the pin to the SPI. Refer to **Table D-37**.

Table D-37 MPAR Pin Assignments

MPAR Field	MPAR Bit	Pin Function
MPA0	0 1	PMC0 MISO
MPA1	0 1	PMC1 MOSI
—[1]	—	PMC2 SCK
MPA3	0 1	PMC3 $\overline{SS}$
—[1]	—	PMC4 RXDB
—[1]	—	PMC5 TXDB
—[1]	—	PMC6 RXDA
—[1]	—	PMC7 TXDA

NOTES:
1. MPA[7:4], MPA2 are not implemented.

Bits [15:8], [7:4], 2 — Not Implemented

SPI pins designated by the MPAR as general-purpose I/O are controlled only by MDDR and PORTMC. The SPI has no effect on these pins. The MPAR does not affect the operation of the SCI submodule.

D.6.7 MCCI Data Direction Register

MDDR — MCCI Data Direction Register **$YFFC0A**

15 … 8	7	6	5	4	3	2	1	0
NOT USED	DDR7	DDR6	DDR5	DDR4	DDR3	DDR2	DDR1	DDR0
RESET:	0	0	0	0	0	0	0	0

MDDR determines whether pins configured for general purpose I/O are inputs or outputs. MDDR affects both SPI function and I/O function. During reset, all MCCI pins are configured as inputs. **Table D-38** shows the effect of MDDR on MCCI pin function.

Table D-38 Effect of MDDR on MCCI Pin Function

MCCI Pin	Mode	MDDR Bit	Bit State	Pin Function
MISO	Master	DDR0	0	Serial data input to SPI
			1	Disables data input
	Slave		0	Disables data output
			1	Serial data output from SPI
MOSI	Master	DDR1	0	Disables data output
			1	Serial data output from SPI
	Slave		0	Serial data input to SPI
			1	Disables data input
SCK[1]	Master	DDR2	—	Clock output from SPI
	Slave		—	Clock input to SPI
SS	Master	DDR3	0	Assertion causes mode fault
			1	General purpose I/O
	Slave		0	SPI slave-select input
			1	Disables slave-select input
RXDB[2]	—	DDR4	0	General purpose I/O
			1	Serial data input to SCIB
TXDB[3]	—	DDR5	0	General purpose I/O
			1	Serial data output from SCIB
RXDA	—	DDR6	0	General purpose I/O
			1	Serial data input to SCIA
TXDA[3]	—	DDR7	0	General purpose I/O
			1	Serial data output from SCIA

NOTES:
1. SCK is automatically assigned to the SPI whenever the SPI is enabled (when the SPE bit in the SPCR1 is set).
2. PMC4 and PMC6 function as general purpose I/O pins when the corresponding RE bit in the SCI control register (SCCR0A or SCCR0B) is cleared.
3. PMC5 and PMC7 function as general purpose I/O pins when the corresponding TE bit in the SCI control register (SCCR0A or SCCR0B) is cleared.

D.6.8 MCCI Port Data Registers

PORTMC — MCCI Port Data Register **$YFFC0C**
PORTMCP — MCCI Port Pin State Register **$YFFC0E**

15 … 9	8	7	6	5	4	3	2	1	0
NOT USED	PMC7	PMC6	PMC5	PMC4	PMC3	PMC2	PMC1	PMC0	
RESET:									
0 … 0	U	U	U	U	U	U	U	U	

Two registers are associated with port MCCI, the MCCI general-purpose I/O port. Pins used for general-purpose I/O must be configured for that function. When using port MCCI as an output port, after configuring the pins as I/O, write the first byte to be output before writing to the MDDR. Afterwards, write to the MDDR to assign each I/O pin as either input or output. This outputs the value contained in register PORTMC for all pins defined as outputs. To output different data, write another byte to PORTMC.

Writes to PORTMC are stored in the internal data latch. If any bit of PORTMC is configured as discrete output, the value latched for that bit is driven onto the pin. Reads of PORTMC return the value of the pin only if the pin is configured as a discrete input. Otherwise, the value read is the value of the latch.

Reads of PORTMCP always return the state of the pins regardless of whether the pins are configured for input or output. Writes to PORTMCP have no effect.

D.6.9 SCI Control Register 0

SCCR0A — SCIA Control Register 0 **$YFFC18**
SCCR0B — SCIB Control Register 0 **$YFFC28**

15 … 13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED	SCBR[12:0]												
RESET:													
0 … 0	0	0	0	0	0	0	0	0	0	1	0	0	

SCCR0 contains the SCI baud rate selection field. Baud rate must be set before the SCI is enabled. The CPU16 can read and write SCCR0 at any time. Changing the value of SCCR0 bits during a transfer operation can disrupt the transfer.

Bits [15:13] — Not Implemented

SCBR[12:0] — SCI Baud Rate

SCI baud rate is programmed by writing a 13-bit value to this field. Writing a value of zero to SCBR disables the baud rate generator. Baud clock rate is calculated as follows:

$$\text{SCI Baud Rate} = \frac{f_{sys}}{32 \times \text{SCBR[12:0]}}$$

or

$$\text{SCBR[12:0]} = \frac{f_{sys}}{32 \times \text{SCI Baud Rate Desired}}$$

where SCBR[12:0] is in the range of 1 to 8191. Writing a value of zero to SCBR disables the baud rate generator. There are 8191 different bauds available. The baud value depends on the value for SCBR and the system clock, as used in the equation above. **Table D-39** shows possible baud rates for a 16.78 MHz system clock. The maximum baud rate with this system clock speed is 524 kbaud.

Table D-39 Examples of SCI Baud Rates

Nominal Baud Rate	Actual Baud Rate	Percent Error	Value of SCBR
500,00.00	524,288.00	4.86	1
38,400.00	37,449.14	–2.48	14
32,768.00	32,768.00	0.00	16
19,200.00	19,418.07	1.14	27
9,600.00	9,532.51	–0.70	55
4,800.00	4,809.98	0.21	109
2,400.00	2,404.99	0.21	218
1,200.00	1,199.74	–0.02	437
600.00	599.87	–0.02	874
300.00	299.94	–0.02	1,748
110.00	110.01	0.01	4,766
64.00	64.00	0.01	8,191

D.6.10 SCI Control Register 1

SCCR1A — SCIA Control Register 1 **$YFFC1A**
SCCR1B — SCIB Control Register 1 **$YFFC2A**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED	LOOPS	WOMS	ILT	PT	PE	M	WAKE	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
RESET:															
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SCCR1 contains SCI configuration parameters, including transmitter and receiver enable bits, interrupt enable bits, and operating mode enable bits. SCCR0 can be read or written at any time. The SCI can modify the RWU bit under certain circumstances. Changing the value of SCCR1 bits during a transfer operation can disrupt the transfer.

Bit 15 — Not Implemented

LOOPS — Loop Mode

0 = Normal SCI operation, no looping, feedback path disabled.
1 = Test SCI operation, looping, feedback path enabled.

The LOOPS bit in SCCR1 controls a feedback path on the data serial shifter. When LOOPS is set, SCI transmitter output is fed back into the receive serial shifter. The TXD pin is asserted (idle line). Both transmitter and receiver must be enabled prior to entering loop mode.

WOMS — Wired-OR Mode for SCI Pins
0 = If configured as an output, TXD is a normal CMOS output.
1 = If configured as an output, TXD is an open-drain output.

ILT — Idle-Line Detect Type
0 = Short idle-line detect (start count on first one).
1 = Long idle-line detect (start count on first one after stop bit(s)).

PT — Parity Type
0 = Even parity
1 = Odd parity

PE — Parity Enable
0 = SCI parity disabled.
1 = SCI parity enabled.

M — Mode Select
0 = 10-bit SCI frame — 1 start bit, 8 data bits, 1 stop bit.
1 = 11-bit SCI frame — 1 start bit, 9 data bits, 1 stop bit.

WAKE — Wakeup by Address Mark
0 = SCI receiver awakened by idle-line detection.
1 = SCI receiver awakened by address mark (last data bit set).

TIE — Transmit Interrupt Enable
0 = SCI TDRE interrupts disabled.
1 = SCI TDRE interrupts enabled.

TCIE — Transmit Complete Interrupt Enable
0 = SCI TC interrupts disabled.
1 = SCI TC interrupts enabled.

RIE — Receiver Interrupt Enable
0 = SCI RDRF and OR interrupts disabled.
1 = SCI RDRF and OR interrupts enabled.

ILIE — Idle-Line Interrupt Enable
0 = SCI IDLE interrupts disabled.
1 = SCI IDLE interrupts enabled.

TE — Transmitter Enable
0 = SCI transmitter disabled (TXD pin can be used as I/O).
1 = SCI transmitter enabled (TXD pin dedicated to SCI transmitter).

RE — Receiver Enable
0 = SCI receiver disabled.
1 = SCI receiver enabled.

RWU — Receiver Wakeup
0 = Normal receiver operation (received data recognized).
1 = Wakeup mode enabled (received data ignored until receiver is awakened).

SBK — Send Break
0 = Normal operation
1 = Break frame(s) transmitted after completion of the current frame.

D.6.11 SCI Status Register

SCSRA — SCIA Status Register **$YFFC1C**
SCSRB — SCIB Status Register **$YFFC2C**

15 … 9	8	7	6	5	4	3	2	1	0
NOT USED	TDRE	TC	RDRF	RAF	IDLE	OR	NF	FE	PF
RESET:	1	1	0	0	0	0	0	0	0

SCSR contains flags that show SCI operating conditions. These flags are cleared either by SCI hardware or by a read/write sequence. The sequence consists of reading SCSR, then reading or writing SCDR.

If an internal SCI signal for setting a status bit comes after reading the asserted status bits, but before writing or reading SCDR, the newly set status bit is not cleared. SCSR must be read again with the bit set and SCDR must be read or written before the status bit is cleared.

A long-word read can consecutively access both SCSR and SCDR. This action clears receive status flag bits that were set at the time of the read, but does not clear TDRE or TC flags. Reading either byte of SCSR causes all 16 bits to be accessed, and any status bit already set in either byte is cleared on a subsequent read or write of SCDR.

Bits [15:9] — Not Implemented

TDRE — Transmit Data Register Empty
0 = Transmit data register still contains data to be sent to the transmit serial shifter.
1 = A new character can now be written to the transmit data register.

TC — Transmit Complete
0 = SCI transmitter is busy.
1 = SCI transmitter is idle.

RDRF — Receive Data Register Full
0 = Receive data register is empty or contains previously read data.
1 = Receive data register contains new data.

RAF — Receiver Active

0 = SCI receiver is idle.

1 = SCI receiver is busy.

IDLE — Idle-Line Detected

0 = SCI receiver did not detect an idle-line condition.

1 = SCI receiver detected an idle-line condition.

OR — Overrun Error

0 = Receive data register is empty and can accept data from the receive serial shifter.

1 = Receive data register is full and cannot accept data from the receive serial shifter. Any data in the shifter is lost and RDRF remains set.

NF — Noise Error

0 = No noise detected in the received data.

1 = Noise detected in the received data.

FE — Framing Error

0 = No framing error detected in the received data.

1 = Framing error or break detected in the received data.

PF — Parity Error

0 = No parity error detected in the received data.

1 = Parity error detected in the received data.

D.6.12 SCI Data Register

SCDRA — SCIA Data Register **$YFFC1E**

SCDRB — SCIB Data Register **$YFFC2E**

15 … 9	8	7	6	5	4	3	2	1	0
NOT USED	R8/T8	R7/T7	R6/T6	R5/T5	R4/T4	R3/T3	R2/T2	R1/T1	R0/T0
RESET:	U	U	U	U	U	U	U	U	U

SCDR consists of two data registers located at the same address. The receive data register (RDR) is a read-only register that contains data received by the SCI serial interface . Data comes into the receive serial shifter and is transferred to RDR. The transmit data register (TDR) is a write-only register that contains data to be transmitted. Data is first written to TDR, then transferred to the transmit serial shifter, where additional format bits are added before transmission. R[7:0]/T[7:0] contain either the first eight data bits received when SCDR is read, or the first eight data bits to be transmitted when SCDR is written. R8/T8 are used when the SCI is configured for nine-bit operation. When the SCI is configured for 8-bit operation, R8/T8 have no meaning or effect.

D.6.13 SPI Control Register

SPCR — SPI Control Register **$YFFC38**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPIE	SPE	WOMP	MSTR	CPOL	CPHA	LSBF	SIZE	SPBR[7:0]							
RESET:															
0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0

The SPCR contains parameters for configuring the SPI. The register can be read or written at any time.

SPIE — SPI Interrupt Enable
- 0 = SPI interrupts disabled.
- 1 = SPI interrupts enabled.

SPE — SPI Enable
- 0 = SPI is disabled.
- 1 = SPI is enabled.

WOMP — Wired-OR Mode for SPI Pins
- 0 = Outputs have normal CMOS drivers.
- 1 = Pins designated for output by MDDR have open-drain drivers, regardless of whether the pins are used as SPI outputs or for general-purpose I/O, and regardless of whether the SPI is enabled.

MSTR — Master/Slave Mode Select
- 0 = SPI is a slave device.
- 1 = SPI is system master.

CPOL — Clock Polarity
- 0 = The inactive state value of SCK is logic level zero.
- 1 = The inactive state value of SCK is logic level one.

CPOL is used to determine the inactive state of the serial clock (SCK). It is used with CPHA to produce a desired clock/data relationship between master and slave devices.

CPHA — Clock Phase
- 0 = Data captured on the leading edge of SCK and changed on the trailing edge of SCK.
- 1 = Data is changed on the leading edge of SCK and captured on the trailing edge of SCK.

CPHA determines which edge of SCK causes data to change and which edge causes data to be captured. CPHA is used with CPOL to produce a desired clock/data relationship between master and slave devices.

LSBF — Least Significant Bit First
- 0 = Serial data transfer starts with LSB.
- 1 = Serial data transfer starts with MSB.

SIZE — Transfer Data Size

0 = 8-bit data transfer.

1 = 16-bit data transfer.

SPBR[7:0] — Serial Clock Baud Rate

The SPI uses a modulus counter to derive the SCK baud rate from the MCU system clock. Baud rate is selected by writing a value from 2 to 255 into SPBR[7:0].

The following expressions apply to SCK baud rate:

$$\text{SCK Baud Rate} = \frac{f_{sys}}{2 \times \text{SPBR[7:0]}}$$

or

$$\text{SPBR[7:0]} = \frac{f_{sys}}{2 \times \text{SCK Baud Rate Desired}}$$

Giving SPBR[7:0] a value of zero or one disables SCK (disable state determined by CPOL). At reset, the SCK baud rate is initialized to one-eighth of the system clock frequency.

D.6.14 SPI Status Register

SPSR — SPI Status Register **$YFFC3C**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPIF	WCOL	0	MODF	0	0	0	0	0	0	0	0	0	0	0	0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPSR contains information concerning the current serial transmission. Only the SPI can set bits in SPSR. The CPU16 reads SPSR to obtain SPI status information and writes it to clear status flags.

SPIF — SPI Finished Flag

0 = SPI is not finished.

1 = SPI is finished.

WCOL — Write Collision

0 = No attempt to write to the SPDR happened during the serial transfer.

1 = Write collision occurred.

Clearing WCOL is accomplished by reading the SPSR while WCOL is set and then either reading the SPDR prior to SPIF being set, or reading or writing the SPDR after SPIF is set.

MODF — Mode Fault Flag

0 = Normal operation.

1 = Another SPI node requested to become the network SPI master while the SPI was enabled in master mode ($\overline{SS}$ input taken low).

The SPI asserts MODF when the SPI is in master mode (MSTR = 1) and the $\overline{SS}$ input pin is negated by an external driver.

D.6.15 SPI Data Register

SPDR — SPI Data Register **$YFFC3E**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UPPB[7:0]								LOWB[7:0]							
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

UPPB — Upper Byte

In 16-bit transfer mode, the upper byte is contains the most significant 8 bits of the transmitted or received data. Bit 15 of the SPDR is the MSB of the 16-bit data.

LOWB — Lower Byte

In 8-bit transfer mode, the lower byte contains the transmitted or received data. MSB in 8-bit transfer mode is bit 7 of the SPDR. In 16-bit transfer mode, the lower byte holds the least significant 8 bits of the data.

D.7 Configurable Timer Module

Table D-40 shows the address map for the CTM7 and the CTM8. The MC68HC16R1/916R1 has the CTM7. The MC68HC16R3/916R3 has the CTM8. In **Table D-40**, shaded areas indicate address locations specific to a particular CTM module.

NOTE

The CTM7 and the CTM8 are referred to as “CTM” except when specific differences require separate identification.

Table D-40 CTM Address Map

Address[1]	15 0	Module
$YFF900	BIUSM Module Configuration Register (BIUMCR)	CTM
$YFF902	BIUSM Test Register (BIUTEST)	
$YFF904	BIUSM Time Base Register (BIUTBR)	
$YFF906	Reserved	
$YFF908	CPSM Control Register (CPCR)	
$YFF90A	CPSM Test Register (CPTR)	
$YFF90C – $YFF90E	Reserved	
$YFF910	MCSM2 Status/Interrupt/Control Register (MCSM2SIC)	
$YFF912	MCSM2 Counter (MCSM2CNT)	
$YFF914	MCSM2 Modulus Latch (MCSM2ML)	
$YFF916	Reserved	
$YFF918	FCSM3 Status/Interrupt/Control Register (FCSM3SIC)	
$YFF91A	FCSM3 Counter (FCSM3CNT)	
$YFF91C – $YFF91E	Reserved	
$YFF920	DASM4 Status/Interrupt/Control Register (DASM4SIC)	CTM7
$YFF922	DASM4 Register A (DASM4A)	
$YFF924	DASM4 Register B (DASM4B)	
$YFF926	Reserved	
$YFF928	DASM5 Status/Interrupt/Control Register (DASM5SIC)	
$YFF92A	DASM5 Register A (DASM5A)	
$YFF92C	DASM5 Register B (DASM5B)	
$YFF92E	Reserved	
$YFF930	SASM6 Status/Interrupt/Control Register A (SIC6A)	CTM
$YFF932	SASM6 Data Register A (S6DATA)	
$YFF934	SASM6 Status/Interrupt/Control Register B (SIC6B)	
$YFF936	SASM6 Data Register B (S6DATB)	
$YFF938 – $YFF93E	Reserved	
$YFF940	SASM8 Status/Interrupt/Control Register A (SIC8A)	
$YFF942	SASM8 Data Register A (S8DATA)	
$YFF944	SASM8 Status/Interrupt/Control Register B (SIC8B)	
$YFF946	SASM8 Data Register B (S8DATB)	
$YFF948 – $YFF94E	Reserved	
$YFF950	SASM10 Status/Interrupt/Control Register A (SIC10A)	

Table D-40 CTM Address Map

Address[1]	15 0	Module
$YFF952	SASM10 Data Register A (S10DATA)	CTM
$YFF954	SASM10 Status/Interrupt/Control Register B (SIC10B)	
$YFF956	SASM10 Data Register B (S10DATB)	
$YFF958 – $YFF95E	Reserved	
$YFF960	SASM12 Status/Interrupt/Control Register A (SIC12A)	
$YFF962	SASM12 Data Register A (S12DATA)	
$YFF964	SASM12 Status/Interrupt/Control Register B (SIC12B)	
$YFF966	SASM12 Data Register B (S12DATB)	
$YFF968 – $YFF96E	Reserved	
$YFF970	SASM14 Status/Interrupt/Control Register A (SIC14A)	
$YFF972	SASM14 Data Register A (S14DATA)	
$YFF974	SASM14 Status/Interrupt/Control Register B (SIC14B)	
$YFF976	SASM14 Data Register B (S14DATB)	
$YFF978 – $YFF97E	Reserved	
$YFF980	SASM16 Status/Interrupt/Control Register A (SIC16A)	
$YFF982	SASM16 Data Register A (S16DATA)	
$YFF984	SASM16 Status/Interrupt/Control Register B (SIC16B)	
$YFF986	SASM16 Data Register B (S16DATB)	
$YFF988 – $YFF98E	Reserved	
$YFF990	PWMSM18 Status/Interrupt/Control Register (PWM18SIC)	
$YFF992	PWMSM18 Period (PWM18A)	
$YFF994	PWMSM18 Pulse Width (PWM18B)	
$YFF996	PWMSM18 Counter (PWM18C)	
$YFF998	PWMSM19 Status/Interrupt/Control Register (PWM19SIC)	
$YFF99A	PWMSM19 Period (PWM19A)	
$YFF99C	PWMSM19 Pulse Width (PWM19B)	
$YFF99E	PWMSM19 Counter (PWM19C)	
$YFF9A0 - $YFF9FF	Reserved	CTM7
$YFF9A0	DASM20 Status/Interrupt/Control Register (DASM20SIC)	CTM8
$YFF9A2	DASM20 Register A (DASM20A)	
$YFF9A4	DASM20 Register B (DASM20B)	
$YFF9A6	Reserved	
$YFF9A8	MCSM21 Status/Interrupt/Control Register (MCSM21SIC)	
$YFF9AA	MCSM21 Counter (MCSM21CNT)	
$YFF9AC	MCSM21 Modulus Latch (MCSM21ML)	
$YFF9AE	Reserved	
$YFF9B0	DASM22 Status/Interrupt/Control Register (DASM22SIC)	
$YFF9B2	DASM22 Register A (DASM22A)	
$YFF9B4	DASM22 Register B (DASM22B)	
$YFF9B6	Reserved	
$YFF9B8	MCSM23 Status/Interrupt/Control Register (MCSM23SIC)	
$YFF9BA	MCSM23 Counter (MCSM23CNT)	

Table D-40 CTM Address Map

Address[1]	15 0	Module
$YFF9BC	MCSM23 Modulus Latch (MCSM23ML)	CTM8
$YFF9BE	Reserved	
$YFF9C0 – $YFF9FF	Reserved	

NOTES:
1. Y = M111, where M is the logic state of the module mapping (MM) bit in the SCIMCR.

D.7.1 BIU Module Configuration Register

BIUMCR — BIU Module Configuration Register **$YFF900**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STOP	FRZ	NOT USED	VECT[7:6]		IARB[2:0]			NOT USED		TBRS 1	NOT USED				TBRS 0
RESET:															
0	0		1	1	0	0	0			0					0

STOP — Low-Power Stop Mode Enable

When the STOP bit is set, the clock to the CTM is shutdown, placing the module into low-power stop mode. The BIUSM still operates in low-power stop mode, allowing the submodule control and data registers to be accessed.

0 = Enable CTM clocks.
1 = Disable CTM clocks.

FRZ — FREEZE Assertion Response

The FRZ bit controls CTM response to assertion of the IMB FREEZE signal. Since the BIUSM propagates FREEZE to the CTM submodules via the submodule bus, the setting of FRZ affects all CTM submodules.

0 = CTM ignores the IMB FREEZE signal.
1 = CTM submodules freeze when the IMB FREEZE signal is asserted.

VECT[7:6] — Interrupt Vector Base Number

This bit field selects the base interrupt vector number for the CTM. Of the eight bits necessary for a vector number, the six low-order bits are hardware defined on a submodule basis, while the two remaining bits are provided by VECT[7:6]. This places the CTM vectors in one of four possible positions in the interrupt vector table. Refer to **Table D-41**.

Table D-41 Interrupt Vector Base Number Bit Field

VECT7	VECT6	Resulting Base Vector Number
0	0	$00
0	1	$40
1	0	$80
1	1	$C0

IARB[2:0] — Interrupt Arbitration Identification ID

This bit field and the IARB3 bit within each submodule capable of requesting interrupts determine the arbitration identification numbers for each submodule requesting interrupt service.

TBRS1, TBRS0 — Time Base Register Bus Select Bits

These bits specify which time base bus is accessed when the time base register (BIUTBR) is read. Refer to **Table D-42**.

Table D-42 Time Base Register Bus Select Bits

TBRS1	TBRS0	Time Base Bus
0	0	TBB1
0	1	TBB2
1	0	TBB3
1	1	TBB4

D.7.2 BIUSM Test Configuration Register

BIUTEST — BIUSM Test Configuration Register **$YFF902**

Used during factory test only.

D.7.3 BIUSM Time Base Register

BIUTBR — BIUSM Time Base Register **$YFF904**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIUTBR is a read-only register used to read the value present on one of the time base buses. The time base bus accessed is determined by TBRS1 and TBRS0 in BIUMCR.

D.7.4 CPSM Control Register

CPCR — CPSM Control Register **$YFF908**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED												PRUN	DIV23	PSEL[1:0]	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PRUN — Prescaler Running

The PRUN bit is a read/write control bit that turns the prescaler counter on and off. This bit allows the counters in various CTM submodules to be synchronized.

0 = Prescaler divider is held in reset and is not running.

1 = Prescaler is running.

DIV23 — Divide By 2/Divide By 3

The DIV23 bit is a read/write control bit that selects the division ratio of the first prescaler stage. It may be changed at any time.

0 = First prescaler stage divides by two.

1 = First prescaler stage divides by three.

PSEL[1:0] — Prescaler Division Ratio Select

This bit field selects the division ratio of the programmable prescaler output signal PCLK6. Refer to **Table D-43**.

Table D-43 Prescaler Division Ratio Select Field

Prescaler Control Bits			Prescaler Division Ratio					
DIV23	**PSEL1**	**PSEL0**	**PCLK1**	**PCLK2**	**PCLK3**	**PCLK4**	**PCLK5**	**PCLK6**
0	0	0	2	4	8	16	32	64
0	0	1	2	4	8	16	32	128
0	1	0	2	4	8	16	32	256
0	1	1	2	4	8	16	32	512
1	0	0	3	6	12	24	48	96
1	0	1	3	6	12	24	48	192
1	1	0	3	6	12	24	48	384
1	1	1	3	6	12	24	48	768

D.7.5 CPSM Test Register

CPTR — CPSM Test Register **$YFF90A**

Used during factory test only.

D.7.6 MCSM Status/Interrupt/Control Registers

MCSMxSIC — MCSM Status/Interrupt/Control Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COF	IL[2:0]			IARB3	NOT USED	DRVA	DRVB	IN2	IN1	EDGE N	EDGE P	NOT USED	CLK[2:0]		
RESET:															
U	0	0	0	0	0	0	0	U	U	0	0	0	0	0	0

Table D-44 shows MCSMxSIC address locations in the CTM. (x) represents the submodule number.

Table D-44 MCSMxSIC Registers

Module		Address
CTM7	**CTM8**	
MCSM2SIC	—	$YFF910
—	MCSM21SIC	$YFF9A8
—	MCSM23SIC	$YFF9B8

COF — Counter Overflow Flag

This bit indicates whether or not a counter overflow has occurred. An overflow of the MCSM counter is defined as the transition of the counter from $FFFF to $xxxx, where $xxxx is the value contained in the modulus latch. If the IL[2:0] field is non-zero, an interrupt request is generated when the COF bit is set.

0 = Counter overflow has not occurred
1 = Counter overflow has occurred

This flag bit is set only by hardware and cleared only by software or by system reset. To clear the flag, first read the bit as a one, then write a zero to the bit.

IL[2:0] — Interrupt Level Field

When the MCSM generates an interrupt request, IL[2:0] determines which of the interrupt request signals is asserted. When a request is acknowledged, the CTM compares IL[2:0] to a mask value supplied by the CPU16 to determine whether to respond. IL[2:0] must have a value in the range of $0 (interrupts disabled) to $7 (highest priority).

IARB3 — Interrupt Arbitration Bit 3

This bit and the IARB[2:0] field in BIUMCR are concatenated to determine the interrupt arbitration number for the submodule requesting interrupt service. Refer to **D.7.1 BIU Module Configuration Register** for more information on IARB[2:0].

DRV[A:B] — Drive Time Base Bus

This field controls the connection of the MCSM to time base buses A and B. Refer to **Table D-45**.

Table D-45 Drive Time Base Bus Field

DRVA	DRVB	Bus Selected
0	0	Neither time base bus A nor bus B is driven
0	1	Time base bus B is driven
1	0	Time base bus A is driven
1	1	Both time base bus A and bus B are driven

WARNING

Two time base buses should not be driven at the same time.

IN2 — Clock Input Pin Status

This read-only bit reflects the logic state of the clock input pin CTM2C. Writing to this bit has no effect nor does reset.

IN1 — Modulus Load Input Pin Status

This read-only bit reflects the logic state of the modulus load input pin CTD9. Writing to this bit has no effect nor does reset.

EDGEN, EDGEP — Modulus Load Edge Sensitivity Bits

These read/write control bits select which edge on CTD9 triggers the modulus load input. Refer to **Table D-46**.

Table D-46 Modulus Load Edge Sensitivity Bits

EDGEN	EDGEP	IN1 Edge Detector Sensitivity
0	0	None
0	1	Positive edge only
1	0	Negative edge only
1	1	Positive and negative edge

CLK[2:0] — Counter Clock Select Field

These read/write control bits select one of the six CPSM clock signals (PCLK[1:6]) or one of two external conditions on CTM2C to clock the modulus counter. The maximum frequency of an external clock signal is $f_{sys}/4$. Refer to **Table D-47**.

Table D-47 Counter Clock Select Field

CLK2	CLK1	CLK0	Free Running Counter Clock Source
0	0	0	Prescaler output 1 (/2 or /3)
0	0	1	Prescaler output 2 (/4 or /6)
0	1	0	Prescaler output 3 (/8 or /12)
0	1	1	Prescaler output 4 (/16 or /24)
1	0	0	Prescaler output 5 (/32 or /48)
1	0	1	Prescaler output 6 (/64 to /768)
1	1	0	CTM2C input pin, negative edge
1	1	1	CTM2C input pin, positive edge

D.7.7 MCSM Counter Registers

MCSMxCNT — MCSM Counter Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table D-48 shows MCSMxCNT address locations in the CTM. (x) represents the sub-module number.

Table D-48 MCSMxCNT Registers

Module		Address
CTM7	CTM8	
MCSM2CNT	—	$YFF912
—	MCSM21CNT	$YFF9AA
—	MCSM23CNT	$YFF9BA

The MCSM counter register is a read/write register. A read returns the current value of the counter. A write simultaneously loads both the counter and the MCSM modulus latch with the specified value. The counter then begins incrementing from this new value.

D.7.8 MCSM Modulus Latch Registers

MCSMxML — MCSM Modulus Latch

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table D-49 shows MCSMxML address locations in the CTM. (x) represents the sub-module number.

Table D-49 MCSMxML Registers

Module		Address
CTM7	**CTM8**	
MCSM2ML	—	$YFF914
—	MCSM21ML	$YFF9AC
—	MCSM23ML	$YFF9BC

The MCSM modulus latch register is a read/write register. A read returns the current value of the latch. A write pre-loads the latch with a new value that the modulus counter will begin counting from when the next load condition occurs.

D.7.9 FCSM Status/Interrupt/Control Register

FCSMSIC — FCSM Status/Interrupt/Control Register **$YFF918**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COF	IL[2:0]			IARB3	NOT USED	DRVA	DRVB	IN	NOT USED				CLK[2:0]		
RESET:															
U	0	0	0	0		0	0	U					0	0	0

COF — Counter Overflow Flag

This flag indicates whether or not a counter overflow has occurred. An overflow is defined as the transition of the counter from $FFFF to $0000. If the IL[2:0] field is non-zero, an interrupt request is generated when the COF bit is set.

0 = Counter overflow has not occurred

1 = Counter overflow has occurred

This flag bit is set only by hardware and cleared by software or system reset. To clear the flag, first read the bit as a one, then write a zero to the bit.

IL[2:0] — Interrupt Level

When the FCSM generates an interrupt request, IL[2:0] determines which of the interrupt request signals is asserted. When a request is acknowledged, the CTM compares IL[2:0] to a mask value supplied by the CPU16 to determine whether to respond. IL[2:0] must have a value in the range of $0 (interrupts disabled) to $7 (highest priority).

IARB3 — Interrupt Arbitration Bit 3

This bit and the IARB[2:0] field in BIUMCR are concatenated to determine the interrupt arbitration number for the submodule requesting interrupt service. Refer to **D.7.1 BIU Module Configuration Register** for more information on IARB[2:0].

DRV[A:B] — Drive Time Base Bus

This field controls the connection of the FCSM to time base buses A and B. Refer to **Table D-50**.

Table D-50 Drive Time Base Bus Field

DRVA	DRVB	Bus Selected
0	0	Neither time base bus A nor bus B is driven
0	1	Time base bus B is driven
1	0	Time base bus A is driven
1	1	Both time base bus A and bus B are driven

WARNING

Two time base buses should not be driven at the same time.

IN — Clock Input Pin Status

This read-only bit reflects the logic state of the clock input pin CTM2C. Writing to this bit has no effect nor does reset.

CLK[2:0] — Counter Clock Select Field

These read/write control bits select one of the six CPSM clock signals (PCLK[1:6]) or one of two external conditions on CTM2C to clock the free-running counter. The maximum frequency of an external clock signal is $f_{sys}/4$. Refer to **Table D-51**.

Table D-51 Counter Clock Select Field

CLK2	CLK1	CLK0	Free Running Counter Clock Source
0	0	0	Prescaler output 1 (/2 or /3)
0	0	1	Prescaler output 2 (/4 or /6)
0	1	0	Prescaler output 3 (/8 or /12)
0	1	1	Prescaler output 4 (/16 or /24)
1	0	0	Prescaler output 5 (/32 or /48)
1	0	1	Prescaler output 6 (/64 or /512 or /96 to /768)
1	1	0	CTM2C input pin, negative edge
1	1	1	CTM2C input pin, positive edge

D.7.10 FCSM Counter Register

FCSMCNT — FCSM Counter Register **$YFF91A**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The FCSM counter register is a read/write register. A read returns the current value of the counter. A write loads the counter with the specified value. The counter then begins incrementing from this new value.

D.7.11 DASM Status/Interrupt/Control Registers

DASMxSIC — DASM Status/Interrupt/Control Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLAG	IL[2:0]			IARB3	NOT USED	WOR	BSL	IN	FORC A	FORC B	ED-POL	MODE[3:0]			
RESET:															
0	0	0	0	0		0	0	U	0	0	0	0	0	0	0

Table D-52 shows DASMxSIC address locations in the CTM. (x) represents the submodule number.

Table D-52 DASMxSIC Registers

Module		Address
CTM7	**CTM8**	
DASM4SIC	—	$YFF920
DASM5SIC		$YFF928
—	DASM20SIC	$YFF9A0
—	DASM22SIC	$YFF9B0

FLAG — Event Flag

This status bit indicates whether or not an input capture or output compare event has occurred. If the IL[2:0] field is non-zero, an interrupt request is generated when the FLAG bit is set.

0 = An input capture or output compare event has not occurred.

1 = An input capture or output compare event has occurred.

Table D-53 shows the status of the FLAG bit in different DASM operating modes.

Table D-53 DASM Mode Flag Status Bit States

Mode	Flag Status Bit State
DIS	FLAG bit is reset
IPWM	FLAG bit is set each time there is a capture on channel A
IPM	FLAG bit is set each time there is a capture on channel A, except for the first time
IC	FLAG bit is set each time there is a capture on channel A
OCB	FLAG bit is set each time there is a successful comparison on channel B
OCAB	FLAG bit is set each time there is a successful comparison on either channel A or B
OPWM	FLAG bit is set each time there is a successful comparison on channel A

The FLAG bit is set by hardware and cleared by software, or by system reset. Clear the FLAG bit either by writing a zero to it, having first read the bit as a one, or by selecting the DIS mode.

IL[2:0] — Interrupt Level

When the DASM generates an interrupt request, IL[2:0] determines which of the interrupt request signals is asserted. When a request is acknowledged, the CTM compares IL[2:0] to a mask value supplied by the CPU16 to determine whether to respond. IL[2:0] must have a value in the range of $0 (interrupts disabled) to $7 (highest priority).

IARB3 — Interrupt Arbitration Bit 3

This bit and the IARB[2:0] field in BIUMCR are concatenated to determine the interrupt arbitration number for the submodule requesting interrupt service. Refer to **D.7.1 BIU Module Configuration Register** for more information on IARB[2:0].

WOR — Wired-OR Mode

In the DIS, IPWM, IPM and IC modes, the WOR bit is not used. Reading this bit returns the value that was previously written.

In the OCB, OCAB and OPWM modes, the WOR bit selects whether the output buffer is configured for open-drain or normal operation.

0 = Output buffer operates in normal mode.
1 = Output buffer operates in open-drain mode.

BSL — Bus Select

This bit selects the time base bus connected to the DASM.

0 = DASM is connected to time base bus A.
1 = DASM is connected to time base bus B.

IN — Input Pin Status

In the DIS, IPWM, IPM and IC modes, this read-only status bit reflects the logic level on the input pin.

In the OCB, OCAB and OPWM modes, reading this bit returns the value latched on the output flip-flop, after EDPOL polarity selection.

Writing to this bit has no effect.

FORCA — Force A

In the OCB, OCAB and OPWM modes, FORCA bit allows software to force the output flip-flop to behave as if a successful comparison had occurred on channel A (except that the FLAG bit is not set). Writing a one to FORCA sets the output flip-flop; writing a zero has no effect.

In the DIS, IPWM, IPM and IC modes, the FORCA bit is not used and writing to it has no effect.

FORCA is cleared by reset, and always reads as zero.

NOTE

Writing a one to both FORCA and FORCB simultaneously resets the output flip-flop.

FORCB — Force B

In the OCB, OCAB and OPWM modes, FORCB allows software to force the output flip-flop to behave as if a successful comparison had occurred on channel B (except that the FLAG bit is not set). Writing a one to FORCB sets the output flip-flop, writing a zero has no effect.

In the DIS, IPWM, IPM and IC modes, the FORCB bit is not used and writing to it has no effect.

FORCB is cleared by reset, and always reads as zero.

NOTE

Writing a one to both FORCA and FORCB simultaneously resets the output flip-flop.

EDPOL — Edge Polarity Bit

EDPOL selects different options depending on the DASM operating mode. Refer to **Table D-54**.

Table D-54 Edge Polarity

MODE	EDPOL	Function
DIS	X	EDPOL is not used in DIS mode
IPWM	0	Channel A captures on a rising edge Channel B captures on a falling edge
	1	Channel A captures on a falling edge Channel B captures on a rising edge
IPM, IC	0	Channel A captures on a rising edge
	1	Channel A captures on a falling edge
OCB, OCAB, OPWM	0	A compare on channel A sets the output pin to logic 1 A compare on channel B clears the output pin to logic 0
	1	A compare on channel A clears the output pin to logic 0 A compare on channel B sets the output pin to logic 1

MODE[3:0] — DASM Mode Select

This bit field selects the mode of operation of the DASM. Refer to **Table D-55**.

NOTE

To avoid spurious interrupts, DASM interrupts should be disabled before changing the operating mode.

Table D-55 DASM Mode Select Field

MODE[3:0]	Bits of Resolution	Time Base Bits Ignored	DASM Operating Mode
0000	—	—	DIS – Disabled
0001	16	—	IPWM – Input pulse width measurement
0010	16	—	IPM – Input measurement period
0011	16	—	IC – Input capture
0100	16	—	OCB – Output compare, flag on B compare
0101	16	—	OCAB – Output compare, flag on A and B compare
011X	—	—	Not used
1000	16	—	OPWM – Output pulse width modulation
1001	15	15	OPWM – Output pulse width modulation
1010	14	[15:14]	OPWM – Output pulse width modulation
1011	13	[15:13]	OPWM – Output pulse width modulation
1100	12	[15:12]	OPWM – Output pulse width modulation
1101	11	[15:11]	OPWM – Output pulse width modulation
1110	9	[15:9]	OPWM – Output pulse width modulation
1111	7	[15:7]	OPWM – Output pulse width modulation

D.7.12 DASM Data Register A

DASMxA — DASM Data Register A

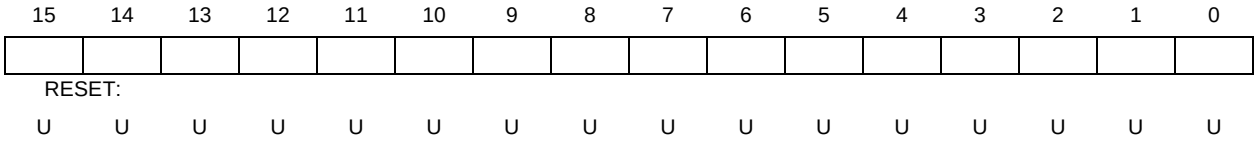


Table D-56 shows DASMxA address locations in the CTM. (x) represents the submodule number.

Table D-56 DASMxA Registers

Module		Address
CTM7	CTM8	
DASM4A	—	$YFF922
DASM5A	—	$YFF92A
—	DASM20A	$YFF9A2
—	DASM22A	$YFF9B2

DASMA is the data register associated with channel A. **Table D-57** shows how DASMA is used with the different modes of operation.

Table D-57 DASMA Operations

Mode	DASMA Operation
DIS	DASMA can be accessed to prepare a value for a subsequent mode selection
IPWM	DASMA contains the captured value corresponding to the trailing edge of the measured pulse
IPM	DASMA contains the captured value corresponding to the most recently detected user-specified rising or falling edge
IC	DASMA contains the captured value corresponding to the most recently detected user-specified rising or falling edge
OCB	DASMA is loaded with the value corresponding to the leading edge of the pulse to be generated. Writing to DASMA in the OCB and OCAB modes also enables the corresponding channel A comparator until the next successful comparison.
OCAB	DASMA is loaded with the value corresponding to the leading edge of the pulse to be generated. Writing to DASMA in the OCB and OCAB modes also enables the corresponding channel A comparator until the next successful comparison.
OPWM	DASMA is loaded with the value corresponding to the leading edge of the PWM pulse to be generated.

D.7.13 DASM Data Register B

DASMxB — DASM Data Register B

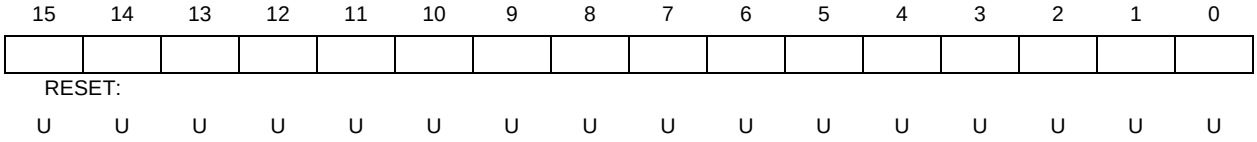


Table D-58 shows DASMxB address locations in the CTM. (x) represents the submodule number.

Table D-58 DASMxB Registers

Module		Address
CTM7	CTM8	
DASM4B	—	$YFF924
DASM5B	—	$YFF92C
—	DASM20B	$YFF9A4
—	DASM22B	$YFF9B4

DASMB is the data register associated with channel B. **Table D-59** shows how DASMB is used with the different modes of operation. Depending on the mode selected, software access is to register B1 or register B2.

Table D-59 DASMB Operations

Mode	DASMB Operation
DIS	DASMB can be accessed to prepare a value for a subsequent mode selection. In this mode, register B1 is accessed in order to prepare a value for the OPWM mode. Unused register B2 is hidden and cannot be read, but is written with the same value as register B1 is written.
IPWM	DASMB contains the captured value corresponding to the trailing edge of the measured pulse. In this mode, register B2 is accessed. Buffer register B1 is hidden and cannot be accessed.
IPM	DASMB contains the captured value corresponding to the most recently detected user-specified rising or falling edge. In this mode, register B2 is accessed. Buffer register B1 is hidden and cannot be accessed.
IC	DASMB contains the captured value corresponding to the most recently detected user-specified rising or falling edge. In this mode, register B2 is accessed. Buffer register B1 is hidden and cannot be accessed.
OCB	DASMB is loaded with the value corresponding to the trailing edge of the pulse to be generated. Writing to DASMB in the OCB and OCAB modes also enables the corresponding channel B comparator until the next successful comparison. In this mode, register B2 is accessed. Buffer register B1 is hidden and cannot be accessed.
OCAB	DASMB is loaded with the value corresponding to the trailing edge of the pulse to be generated. Writing to DASMB in the OCB and OCAB modes also enables the corresponding channel B comparator until the next successful comparison. In this mode, register B2 is accessed. Buffer register B1 is hidden and cannot be accessed.
OPWM	DASMB is loaded with the value corresponding to the trailing edge of the PWM pulse to be generated. In this mode, register B1 is accessed. Buffer register B2 is hidden and cannot be accessed.

D.7.14 SASM Status/Interrupt/Control Registers

SICxA — SASM Status/Interrupt/Control Register A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLAG	IL[2:0]			IARB3	IEN	0	BSL	IN	0	FORCE	ED-OUT	0	0	MODE[1:0]	
RESET:															
0	0	0	0	0	0	0	0	U	0	0	0	0	0	0	0

Table D-60 shows SICxA address locations in the CTM. (x) represents the submodule number.

Table D-60 SICxA Registers

SICxA Register	Address
SIC6A	$YFF930
SIC8A	$YFF940
SIC10A	$YFF950
SIC12A	$YFF960
SIC14A	$YFF970
SIC16A	$YFF980

SICxB — SASM Status/Interrupt/Control Register B

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLAG	0	0	0	0	IEN	0	BSL	IN	0	FORCE	ED-OUT	0	0	MODE[1:0]	
RESET:															
0	0	0	0	0	0	0	0	U	0	0	0	0	0	0	0

Table D-61 shows SICxB address locations in the CTM. (x) represents the submodule number.

Table D-61 SICxB Registers

SICxB Register	Address
SIC6B	$YFF934
SIC8B	$YFF944
SIC10B	$YFF954
SIC12B	$YFF964
SIC14B	$YFF974
SIC16B	$YFF984

SICxA and SICxB contain the control and status bits for SASM channels A and B, respectively. SICxA also contains the IL[2:0] interrupt level field and IARB3 interrupt arbitration bit 3 for both SASM channels A and B.

FLAG – Event Flag

FLAG indicates whether or not an input capture or output compare event has occurred. If the IL[2:0] field is non-zero, and IEN is set, an interrupt request is generated when FLAG is set.

0 = An input capture or output compare event has not occurred.
1 = An input capture or output compare event has occurred.

Table D-62 shows the event flag status during different modes.

Table D-62 Event Flag Status Conditions

Mode	Status Description
IC	If a subsequent input capture event occurs while FLAG is set, the new value is latched and FLAG remains set.
OC	If a subsequent output compare event occurs while FLAG is set, the compare occurs normally and FLAG remains set.
OCT	If a subsequent output compare event occurs while FLAG is set, the output signal toggles normally and FLAG remains set.
OP	If a subsequent internal compare event occurs while FLAG is set, the compare occurs normally and FLAG remains set.

FLAG is set only by hardware and cleared only by software or by a system reset.To clear this bit, first read the register with FLAG set to one, then write a zero to the bit.

NOTE

The flag clearing mechanism works only if no flag setting event occurs between the read and write operations. If a FLAG setting event occurs between the read and write operations, the FLAG bit will not cleared.

IL[2:0] — Interrupt Level

Setting IP[2:0] to a non-zero value causes the SASM to request an interrupt when the FLAG bit sets. If IL[2:0] = %000, no interrupts will be requested when FLAG sets.

NOTE

This field affects both SASM channels, not just channel A.

IARB3 — Interrupt Arbitration Bit 3

This bit works in conjunction with IARB[2:0] in the BIUMCR. Each module that generates interrupt requests on the IMB must have a unique value in the arbitration field. This interrupt arbitration identification number is used to arbitrate for the IMB when modules generate simultaneous interrupts of the same priority. The IARB3 bit is cleared by reset. Refer to **12.4.4 BIUSM Registers** for more information on IARB[2:0].

NOTE

This bit field affects both SASM channels, not just channel A.

IEN — Interrupt Enable

This control bit enables interrupts when FLAG is set and the IL[2:0] field is non-zero.

0 = Interrupts disabled.
1 = Interrupts enabled.

BSL— Time Base Bus Select

This control bit selects the time base bus connected to the SASM.

0 = Time base bus A selected.
1 = Time base bus B selected.

IN — Input Pin Status

In input mode (IC), the IN bit reflects the logic state present on the corresponding input pin after being Schmitt triggered and synchronized.

In the output modes (OC, OCT and OP), the IN bit value reflects the state of the output flip-flop.

The IN bit is a read-only bit. Reset has no effect on this bit.

FORCE — Force Compare Control

In the IC and OP modes, FORCE is not used and writing to it has no effect.

In the OC and OCT modes, FORCE is used by software to cause the output flip-flop (and the output pin) to behave as though an output compare had occurred. In OC and OCT mode, setting FORCE causes the value of EDOUT to be transferred to the output of the output flip-flop. Internal synchronization ensures that the correct level appears on the output pin when a new value is written to EDOUT and FORCE is set at the same time.

0 = No action.

1 = Force output flip-flop to behave as if an output compare has occurred.

FORCE is cleared by reset and always reads as zero.

NOTE

FLAG is not affected by the use of the FORCE bit.

EDOUT — Edge Detect and Output Level

In IC mode, EDOUT is used to select the edge that triggers the input capture circuitry.

0 = Input capture on falling edge.

1 = Input capture on rising edge.

In OC and OCT mode, the EDOUT bit is used to latch the value to be output to the pin on the next output compare match or when the FORCE is set. Internal synchronization ensures that the correct level appears on the output pin when a new value is written to EDOUT and FORCE is set at the same time. Reading EDOUT returns the previous value written.

In OP mode, the value of EDOUT is output to the corresponding pin. Reading EDOUT returns the previous value written.

MODE[1:0] — SASM Operating Mode

This bit field selects the mode of operation for the SASM channel. Refer to **Table D-63**.

Table D-63 SASM Operating Mode Select

MODE1	MODE2	SASM Channel Operating Mode
0	0	Input capture (IC)
0	1	Output port (OP)
1	0	Output compare (OC)
1	1	Output compare and toggle (OCT)

D.7.15 SASM Data Registers

SxDATA — SASM Data Register A

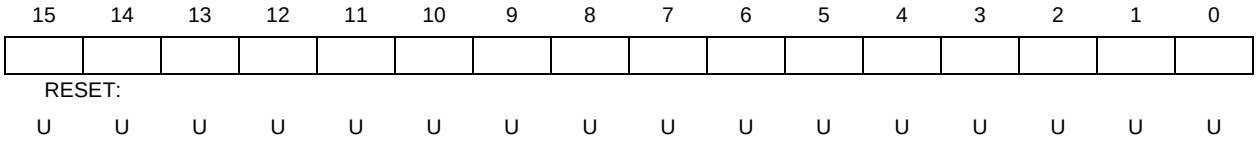


Table D-64 shows SxDATA address locations in the CTM. (x) represents the submodule number.

Table D-64 SxDATA Registers

SxDATA Register	Address
S6DATA	$YFF932
S8DATA	$YFF942
S10DATA	$YFF952
S12DATA	$YFF962
S14DATA	$YFF972
S16DATA	$YFF982

SxDATB — SASM Data Register B

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

Table D-65 shows SxDATB address locations in the CTM. (x) represents the submodule number.

Table D-65 SxDATB Registers

SxDATB Register	Address
S6DATB	$YFF936
S8DATB	$YFF946
S10DATB	$YFF956
S12DATB	$YFF966
S14DATB	$YFF976
S16DATB	$YFF986

SDATA and SDATB are the data registers associated with SASM channels A and B, respectively. In IC mode, SDATA and SDATB contain the last captured value. In the OC, OCT and OP modes, SDATA and SDATB are loaded with the value of the next output compare.

D.7.16 PWM Status/Interrupt/Control Register

PWMxSIC — PWM Status/Interrupt/Control Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLAG	IL[2:0]			IARB3	NOT USED			PIN	NOT USED	LOAD	POL	EN	CLK[2:0]		
RESET:															
0	0	0	0	0				0		0	0	0	0	0	0

Table D-66 shows PWMxSIC address locations in the CTM. (x) represents the submodule number.

Table D-66 PWMxSIC Registers

PWMxSIC Register	Address
PWM18SIC	$YFF990
PWM19SIC	$YFF998

FLAG — Period Completion Status

This status bit indicates when the PWM output period has been completed.

0 = PWM period is not complete.

1 = PWM period is complete.

The FLAG bit is set each time a PWM period is completed. Whenever the PWM is enabled, the FLAG bit is set immediately to indicate that the contents of the buffer registers PWMA2 and PWMB2 have been updated, and that the period using these new values has started. It also indicates that the user accessible period and pulse width registers PWMA1 and PWMB1 can be loaded with values for the next PWM period. Once set, the FLAG bit remains set and is not affected by any subsequent period completions, until it is cleared.

Only software can clear the FLAG bit. To clear FLAG, first read the bit as one then write a zero to the bit. Writing a one to FLAG has no effect. When the PWM is disabled, FLAG remains cleared.

When the interrupt level set specified by IL[2:0] is non-zero, an interrupt request is generated when the FLAG bit is set.

IL[2:0] — Interrupt Level Field

When the PWMSM generates an interrupt request, IL[2:0] determines which of the interrupt request signals is asserted. When a request is acknowledged, the CTM7 compares IL[2:0] to a mask value supplied by the CPU32 to determine whether to respond. IL[2:0] must have a value in the range of $0 (interrupts disabled) to $7 (highest priority).

IARB3 — Interrupt Arbitration Bit 3

This bit and the IARB[2:0] field in BIUMCR are concatenated to determine the interrupt arbitration number for the submodule requesting interrupt service. Refer to **D.7.1 BIU Module Configuration Register** for more information on IARB[2:0].

PIN — Output Pin Status

This status bit indicates the logic state present on the PWM output pin.

0 = Logic zero present on the PWM output pin.

1 = Logic one present on the PWM output pin.

PIN is a read-only bit; writing to it has no effect.

LOAD — Period and Pulse Width Register Load Control

Setting LOAD reinitializes the PWMSM and starts a new PWM period without causing a glitch on the output signal.

0 = No action

1 = Load period and pulse width registers

This bit is always read as a zero. Writing a one to this bit results in the following immediate actions:

- The contents of PWMA1 (period value) are transferred to PWMA2.
- The contents of PWMB1 (pulse width value) are transferred to PWMB2.
- The counter register (PWMC) is initialized to $0001.
- The control logic and state sequencer are reset.
- The FLAG bit is set.
- The output flip-flop is set if the new value in PWMB2 is not $0000.

NOTE

Writing a one to the LOAD bit when the EN bit = 0, (when the PWMSM is disabled), has no effect.

POL — Output Pin Polarity Control

This control bit sets the polarity of the PWM output signal. It works in conjunction with the EN bit and controls whether the PWMSM drives the output pin with the non-inverted or inverted state of the output flip-flop. Refer to **Table D-67**.

Table D-67 PWMSM Output Pin Polarity Selection

POL	EN	Output Pin State	Periodic Edge	Variable Edge	Optional Interrupt On
0	0	Always low	—	—	—
1	0	Always high	—	—	—
0	1	High pulse	Rising edge	Falling edge	Rising edge
1	1	Low pulse	Falling edge	Rising edge	Falling edge

EN — PWMSM Enable

This control bit enables and disables the PWMSM.

0 = Disable the PWMSM.
1 = Enable the PWMSM.

While the PWMSM is disabled (EN = 0):

- The output flip-flop is held in reset and the level on the output pin is set to one or zero according to the state of the POL bit.
- The PWMSM divide-by-256 prescaler is held in reset.
- The counter stops incrementing and is at $0001.
- The comparators are disabled.
- The PWMA1 and PWMB1 registers permanently transfer their contents to the buffer registers PWMA2 and PWMB2, respectively.

When the EN bit is changed from zero to one:

- The output flip-flop is set to start the first pulse.
- The PWMSM divide-by-256 prescaler is released.
- The counter is released and starts to increment from $0001.
- The FLAG bit is set to indicate that PWMA1 and PWMB1 can be updated with new values.

While EN is set, the PWMSM continuously generates a pulse width modulated output signal based on the data in PWMA2 and PWMB2 which are updated via PWMA1 and PWMB2 each time a period is completed.

NOTE

To prevent unwanted output waveform glitches when disabling the PWMSM, first write to PWMB1 to generate one period of 0% duty cycle, then clear EN.

CLK[2:0] — Clock Rate Selection

The CLK[2:0] bits select one of the eight counter clock sources coming from the PWMSM prescaler. These bits can be changed at any time. **Table D-68** shows the counter clock sources and rates in detail.

Table D-68 PWMSM Divide By Options

CLK2	CLK1	CLK0	PCLK1 = f_{sys} ÷ 2 (CPCR DIV23 = 0)	PCLK1 = f_{sys} ÷ 2 (CPCR DIV23 = 0)
0	0	0	f_{sys} ÷ 2	f_{sys} ÷ 3
0	0	1	f_{sys} ÷ 4	f_{sys} ÷ 6
0	1	0	f_{sys} ÷ 8	f_{sys} ÷ 12
0	1	1	f_{sys} ÷ 16	f_{sys} ÷ 24
1	0	0	f_{sys} ÷ 32	f_{sys} ÷ 48
1	0	1	f_{sys} ÷ 64	f_{sys} ÷ 96
1	1	0	f_{sys} ÷ 128	f_{sys} ÷ 192
1	1	1	f_{sys} ÷ 512	f_{sys} ÷ 768

D.7.17 PWM Period Registers

PWMxA1 — PWMA1 Period Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

Table D-69 shows PWMxA1 address locations in the CTM. (x) represents the submodule number.

Table D-69 PWxMA1 Registers

PWMxA1 Register	Address
PWM18A1	$YFF992
PWM19A1	$YFF99A

The PWMA1 register contains the period value for the next cycle of the PWM output waveform. When the PWMSM is enabled, a period value written to PWMxA1 is loaded into PWMA2 at the end of the current period or when the LOAD bit in PWMSIC is written to one. If the PWMSM is disabled, a period value written to PWMA1 is loaded into PWMA2 on the next half cycle of the MCU system clock. PWMA2 is a temporary register that is used to smoothly update the PWM period value; it is not user-accessible. The PWMSM hardware does not modify the contents of PWMA1 at any time.

D.7.18 PWM Pulse Width Registers

PWMxB1 — PWM Pulse Width Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

Table D-69 shows PWMxB1 address locations in the CTM. (x) represents the submodule number.

Table D-70 PWMxB1 Registers

PWMxB Register	Address
PWM18B1	$YFF994
PWM19B1	$YFF99C

The PWMB1 register contains the pulse width value for the next cycle of the PWM output waveform. When the PWMSM is enabled, a pulse width value written to PWMB1 is loaded into PWMB2 at the end of the current period or when the LOAD bit in PWMSIC is written to one. If the PWMSM is disabled, a pulse width value written to PWMB1 is loaded into PWMB2 on the next half cycle of the MCU system clock. PWMB2 is a temporary register that is used to smoothly update the PWM pulse width value; it is not user-accessible. The PWMSM hardware does not modify the contents of PWMB1 at any time.

D.7.19 PWM Counter Registers

PWMxC — PWM Counter Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table D-69 shows PWMxC address locations in the CTM. (x) represents the submodule number.

Table D-71 PWMxC Registers

PWMxC Register	Address
PWM18C	$YFF996
PWM19C	$YFF99E

PWMC holds the current value of the PWMSM counter. PWMC can be read at any time; writing to it has no effect. PWMC is loaded with $0001 on reset and is set and held to that value whenever the PWMSM is disabled.

D.8 Flash EEPROM Modules

The MC68HC916R1 contains two flash EEPROM modules: a 16 K-byte module and a 32-Kbyte module. The MC68HC916R3 has three 32-Kbyte flash EEPROM modules. **Table D-72** shows the flash EEPROM module address map.

Table D-72 Flash EEPROM Address Map

Address[1]	15 0	Module
$YFF800	Flash EEPROM Module Configuration (FEE1MCR)	Flash EEPROM1 (916R1 — 16K) (916R3 — 32K)
$YFF802	Flash EEPROM Test Register (FEE1TST)	
$YFF804	Flash EEPROM Base Address High (FEE1BAH)	
$YFF806	Flash EEPROM Base Address Low (FEE1BAL)	
$YFF808	Flash EEPROM Control Register (FEE1CTL)	
$YFF80A	Reserved	
$YFF80C	Reserved	
$YFF80E	Reserved	
$YFF810	Flash EEPROM Bootstrap Word 0 (FEE1BS0)	
$YFF812	Flash EEPROM Bootstrap Word 1 (FEE1BS1)	
$YFF814	Flash EEPROM Bootstrap Word 2 (FEE1BS2)	
$YFF816	Flash EEPROM Bootstrap Word 3 (FEE1BS3)	
$YFF818	Reserved	
$YFF81A	Reserved	
$YFF81C	Reserved	
$YFF81E	Reserved	
$YFF820	Flash EEPROM Module Configuration (FEE2MCR)	Flash EEPROM2 (916R1 — 32K) (916R3 — 32K)
$YFF822	Flash EEPROM Test Register (FEE2TST)	
$YFF824	Flash EEPROM Base Address High (FEE2BAH)	
$YFF826	Flash EEPROM Base Address Low (FEE2BAL)	
$YFF828	Flash EEPROM Control Register (FEE2CTL)	
$YFF82A	Reserved	
$YFF82C	Reserved	
$YFF82E	Reserved	
$YFF830	Flash EEPROM Bootstrap Word 0 (FEE2BS0)	
$YFF832	Flash EEPROM Bootstrap Word 1 (FEE2BS1)	
$YFF834	Flash EEPROM Bootstrap Word 2 (FEE2BS2)	
$YFF836	Flash EEPROM Bootstrap Word 3 (FEE2BS3)	
$YFF838	Reserved	
$YFF83A	Reserved	
$YFF83C	Reserved	
$YFF83E	Reserved	
$YFF840	Flash EEPROM Module Configuration Register (FEE3MCR)	Flash EEPROM3 (916R3 Only — 32K)
$YFF842	Flash EEPROM Test Register (FEE3TST)	
$YFF844	Flash EEPROM Base Address Register High (FEE3BAH)	
$YFF846	Flash EEPROM Base Address Register Low (FEE3BAL)	
$YFF848	Flash EEPROM Control Register (FEE3CTL)	

Table D-72 Flash EEPROM Address Map

$YFF84A	Reserved	Flash EEPROM3 (916R3 Only — 32K)
$YFF84C	Reserved	
$YFF84E	Reserved	
$YFF850	Flash EEPROM Bootstrap Word 0 (FEE3BS0)	
$YFF852	Flash EEPROM Bootstrap Word 1 (FEE3BS1)	
$YFF854	Flash EEPROM Bootstrap Word 2 (FEE3BS2)	
$YFF856	Flash EEPROM Bootstrap Word 3 (FEE3BS3)	
$YFF858	Reserved	
$YFF85A	Reserved	
$YFF85C	Reserved	
$YFF85E	Reserved	

NOTES:
1. Y = M111, where M is the logic state of the module mapping (MM) bit in the SCIMCR.

NOTE

In the following register diagrams, bits with reset states determined by shadow bits are shaded. The reset value “SB” indicates that a bit assumes the value of its associated shadow bit during reset.

The following register descriptions apply to the corresponding register in all control blocks. References to FEExMCR, for example, apply to FEE1MCR (in the 16-Kbyte module) and FEE2MCR (in the 32-Kbyte module).

D.8.1 Flash EEPROM Module Configuration Register

FEExMCR — Flash EEPROM Module Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STOP	FRZ	0	$\overline{\text{BOOT}}$	LOCK	0	ASPC[1:0]		WAIT[1:0]		0	0	0	0	0	0
RESET:															
$\overline{\text{DATA}}$ $\overline{14}$ + SB	0	0	SB	SB	0	SB	SB	SB	SB	0	0	0	0	0	0

Table D-73 shows FEExMCR address locations.

Table D-73 FEExMCR Registers

916R1 Flash Register	916R3 Flash Register	Address
FEE1MCR	FEE1MCR	$YFF800
FEE2MCR	FEE2MCR	$YFF820
—	FEE3MCR	$YFF840

The FEExMCR register (FEE1MCR, FEE2MCR, FEE3MCR) controls module configuration. This register can be written only when LOCK = 0. All active bits in the FEExMCR take values from the associated shadow register during reset.

STOP — Stop Mode Control

0 = Normal operation.

1 = Low-power stop operation.

STOP can be set either by pulling data bus pin DATA14 low during reset (for all flash EEPROM modules) or by the corresponding shadow bit. The array can be re-enabled by clearing STOP. If STOP is set during programming or erasing, the program/erase voltage is automatically turned off. However, the ENPE control bit in FEExCTL remains set. When STOP is cleared, the program/erase voltage is automatically turned back on if ENPE is set.

FRZ — Freeze Mode Control

0 = Disable program/erase voltage while FREEZE is asserted.

1 = Allow the ENPE bit to turn on the program/erase voltage while FREEZE is asserted.

$\overline{\text{BOOT}}$ — Boot Control

0 = Flash EEPROM module responds to bootstrap addresses after reset.

1 = Flash EEPROM module does not respond to bootstrap addresses after reset.

On reset, $\overline{\text{BOOT}}$ takes on the value stored in its associated shadow bit. If $\overline{\text{BOOT}}$ = 0 and STOP = 0, the module responds to program space accesses to IMB addresses $000000 to $000006 following reset, and the contents of FEExBS[3:0] are used as bootstrap vectors. After address $000006 is read, the module responds normally to control block or array addresses only.

LOCK — Lock Registers

0 = Write-locking disabled.

1 = Write-locked registers protected.

If the reset state of LOCK is zero, it can be set once after reset to allow protection of the registers after initialization. Once the LOCK bit is set by software, it cannot be cleared again until after a reset.

ASPC[1:0] — Flash EEPROM Array Space

ASPC[1:0] assigns the array to supervisor or user space, and to program or data space. The state of ASPC[1:0] out of reset is determined by the value stored in the associated shadow bits. Since the CPU16 runs only in supervisor mode, ASPC1 must remain set to one for array accesses to take place. The field can be written only when LOCK = 0 and STOP = 1. Refer to **Table D-74**.

Table D-74 Array Space Encoding

ASPC[1:0]	Type of Access
10	Supervisor program and data space
11	Supervisor program space

WAIT[1:0] — Wait States

The state of WAIT[1:0] out of reset is determined by the value stored in the associated shadow bits. WAIT[1:0] specifies the number of wait states inserted during accesses to the flash EEPROM module. A wait state has the duration of one system clock cycle. WAIT[1:0] affects both control block and array accesses, and can be written only if LOCK = 0 and STOP = 1. Refer to **Table D-75**.

Table D-75 Wait State Encoding

WAIT[1:0]	Wait States	Clocks Per Transfer
00	0	3
01	1	4
10	2	5
11	–1	2

The value of WAIT[1:0] is compatible with the lower two bits of the $\overline{\text{DSACK}}$ field in the SCIM chip-select option registers. An encoding of %11 in WAIT[1:0] corresponds to an encoding for fast termination.

D.8.2 Flash EEPROM Test Register

FEExTST — Flash EEPROM Test Register

Table D-76 shows FEExTST address locations.

Table D-76 FEExTST Registers

916R1 Flash Register	916R3 Flash Register	Address
FEE1TST	FEE1TST	$YFF802
FEE2TST	FEE2TST	$YFF822
—	FEE3TST	$YFF842

These registers are used for factory test only.

D.8.3 Flash EEPROM Base Address Registers

FEExBAH — Flash EEPROM Base Address Register High

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	ADDR 23	ADDR 22	ADDR 21	ADDR 20	ADDR 19	ADDR 18	ADDR 17	ADDR 16
RESET:															
0	0	0	0	0	0	0	0	SB	SB	SB	SB	SB	SB	SB	SB

Table D-78 shows FEExBAH address locations.

Table D-77 FEExBAH Registers

916R1 Flash Register	916R3 Flash Register	Address
FEE1BAH	FEE1BAH	$YFF804
FEE2BAH	FEE2BAH	$YFF824
—	FEE3BAH	$YFF844

FEExBAL — Flash EEPROM Base Address Register Low (16 Kbyte)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR 15	ADDR 14	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RESET:															
SB	SB	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FEExBAL — Flash EEPROM Base Address Register Low (32 Kbyte)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR 15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RESET:															
SB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table D-78 shows FEExBAL address locations.

Table D-78 FEExBAL Registers

916R1 Flash Register	916R3 Flash Register	Address
FEE1BAL	FEE1BAL	$YFF806
FEE2BAL	FEE2BAL	$YFF826
—	FEE3BAL	$YFF846

The base address high registers (FEE1BAH, FEE2BAH, FEE3BAH) contain the 8 high-order bits of the array base address; the base address low registers (FEE1BAL, FEE2BAL, FEE3BAL) contain the active low-order bit of the array base address. During reset, both FEExBAH and FEExBAL take on default values programmed into associated shadow registers. After reset, if LOCK = 0 and STOP = 1, software can write to FEExBAH and FEExBAL to relocate the array.

D.8.4 Flash EEPROM Control Register

FEExCTL — Flash EEPROM Control Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	VFPE	ERAS	LAT	ENPE
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table D-79 shows FEExCTL address locations.

Table D-79 FEExCTL Registers

916R1 Flash Register	916R3 Flash Register	Address
FEE1CTL	FEE1CTL	$YFF808
FEE2CTL	FEE2CTL	$YFF828
—	FEE3CTL	$YFF848

The FEExCTL register (FEE1CTL, FEE2CTL, FEE3CTL) controls programming and erasure of the arrays. FEExCTL is accessible in supervisor mode only.

VFPE — Verify Program/Erase

0 = Normal read cycles.

1 = Invoke program verify circuit.

The VFPE bit invokes a special program-verify circuit. During programming sequences (ERAS = 0), VFPE is used in conjunction with the LAT bit to determine when programming of a location is complete. If VFPE and LAT are both set, a bit-wise exclusive-OR of the latched data with the data in the location being programmed occurs when any valid FLASH location is read. If the location is completely programmed, a value of zero is read. Any other value indicates that the location is not fully programmed. When VFPE is cleared, normal reads of valid FLASH locations occur. The value of VFPE cannot be changed while ENPE = 1.

ERAS — Erase Control

0 = Flash EEPROM configured for programming

1 = Flash EEPROM configured for erasure

The ERAS bit configures the array for either programming or erasure. Setting ERAS causes all locations in the array and all control bits in the control block to be configured for erasure at the same time.

When the LAT bit is set, ERAS also determines whether a read returns the data in the addressed location (ERAS = 1) or the address itself (ERAS = 0). ERAS cannot be changed while ENPE = 1.

LAT — Latch Control

0 = Programming latches disabled.

1 = Programming latches enabled.

The LAT bit configures the EEPROM array for normal reads or for programming. When LAT is cleared, the FLASH module address and data buses are connected to the IMB address and data buses and the module is configured for normal reads. When LAT is set, module address and data buses are connected to parallel internal latches and the array is configured for programming or erasing.

Once LAT is set, the next write to a valid FLASH module address causes the programming circuitry to latch both address and data. Unless control register shadow bits are to be programmed, the write must be to an array address. The value of LAT cannot be changed while ENPE = 1.

ENPE — Enable Programming/Erase

0 = Disable program/erase voltage.

1 = Apply program/erase voltage to flash EEPROM.

Setting the ENPE bit applies the program/erase voltage to the array. ENPE can be set only after LAT has been set and a write to the data and address latches has occurred. ENPE remains cleared if these conditions are not met. While ENPE is set, the LAT, VFPE, and ERAS bits cannot be changed, and attempts to read an array location are ignored.

D.8.5 Flash EEPROM Bootstrap Words

FEExBS[3:0] — Flash EEPROM Bootstrap Words

Table D-80 shows FEExBS address locations.

Table D-80 FEExBS[3:0] Words

916R1 Flash Bootstrap Words	916R3 Flash Bootstrap Words	Address
FEE1BS[3:0]	FEE1BS[3:0]	$YFF810 - $YFF816
FEE2BS[3:0]	FEE2BS[3:0]	$YFF820 - $YFF826
—	FEE3BS[3:0]	$YFF850 - $YFF856

The flash EEPROM bootstrap words (FEE1BS[3:0], FEE2BS[3:0], FEE3BS[3:0]) can be used as system bootstrap vectors. When BOOT = 1 in FEExMCR during reset, the flash module responds to program space accesses of IMB addresses $000000 to $000006 after reset. When BOOT = 0, the flash module responds only to normal array and register accesses. FEExBS[3:0] can be read at any time, but it can only be changed by programming the appropriate locations. **Table D-81** shows bootstrap word addresses in program space.

Table D-81 Bootstrap Words

Bootstrap Word	Corresponding Boot Address	Corresponding Vector Content
FEE1BS0, FEE2BS0, FEE3BS0	$000000	Initial ZK, SK, and PC
FEE1BS1, FEE2BS1, FEE2BS1	$000002	Initial PC
FEE1BS2, FEE2BS2, FEE3BS2	$000004	Initial SP
FEE1BS3, FEE2BS3, FEE3BS3	$000006	Initial IZ

D.9 Block Erasable Flash

M68HC16 R-series MCUs contain a 2-Kbyte BEFLASH. **Table D-82** shows the BEFLASH address map.

Table D-82 BEFLASH Address Map

Address[1]	15 0
$YFF7A0	BEFLASH Module Configuration Register (BFEMCR)
$YFF7A2	BEFLASH Test Register (BFETST)
$YFF7A4	BEFLASH Base Address High Register (BFEBAH)
$YFF7A6	BEFLASH Base Address Low Register (BFEBAL)
$YFF7A8	BEFLASH Control Register (BFECTL)
$YFF7AA	Reserved
$YFF7AC	Reserved
$YFF7AE	Reserved
$YFF7B0	BEFLASH Bootstrap Word 0 (BFEBS0)
$YFF7B2	BEFLASH Bootstrap Word 1 (BFEBS1)
$YFF7B4	BEFLASH Bootstrap Word 2 (BFEBS2)
$YFF7B6	BEFLASH Bootstrap Word 3 (BFEBS3)
$YFF7B8	Reserved
$YFF7BA	Reserved
$YFF7BC	Reserved
$YFF7BE	Reserved

NOTES:
1. Y = M111, where M is the logic state of the module mapping (MM) bit in the SCIMCR.

NOTE

In the following register diagrams, the reset value SB indicates that a bit assumes the value of its associated shadow bit during reset.

D.9.1 BEFLASH Module Configuration Register

BFEMCR — BEFLASH Module Configuration Register **$YFF7A0**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STOP	FRZ	0	$\overline{\text{BOOT}}$	LOCK	0	ASPC[1:0]		0	0	0	0	0	0	0	0
RESET:															
$\overline{\text{DATA15}}$ + SB	0	0	SB	SB	0	SB	SB	0	0	0	0	0	0	0	0

This register can be written only when the control block is not write-locked (when LOCK = 0). All active bits take values from the associated shadow register during reset.

STOP — Stop Mode Control
0 = Normal operation.
1 = Low-power stop operation.

STOP can be set either by pulling data bus pin DATA15 low during reset or by the corresponding shadow bit. The EEPROM array is inaccessible during low-power stop. The array can be re-enabled by clearing STOP. If STOP is set during programming or erasing, the program/erase voltage is automatically turned off. However, the enable program/erase bit (ENPE) remains set. If STOP is cleared, program/erase voltage is automatically turned back on unless ENPE is cleared.

FRZ — Freeze Mode Control

0 = Disable program/erase voltage while FREEZE is asserted.

1 = Allow ENPE bit to turn on the program/erase voltage while FREEZE signal is asserted.

$\overline{\text{BOOT}}$ — Boot Control

0 = BEFLASH responds to bootstrap vector addresses after reset.

1 = BEFLASH does not respond to bootstrap vector addresses after reset.

On reset, $\overline{\text{BOOT}}$ takes on the value stored in its associated shadow bit. If $\overline{\text{BOOT}}$ = 0 and STOP = 0, the module responds to program space accesses of IMB addresses $000000 to $000006 following reset, and the contents of BFEBS[3:0] are used as bootstrap vectors. After address $000006 is read, the module responds normally to control block or array addresses only.

LOCK — Lock Registers

0 = Write-locking disabled

1 = Write-locked registers protected

If the reset state of the LOCK is zero, it can be set once to protect the registers after initialization. When set, LOCK cannot be cleared until reset occurs.

ASPC[1:0] — BEFLASH Array Space

The CPU16 operates only in supervisory mode, and as a result, ASPC1 must remain set to one for array accesses to take place. The field can be written only if LOCK = 0 and STOP = 1. During reset, ASPC[1:0] takes on the default value programmed into the associated shadow register. Refer to **Table D-83**.

Table D-83 Array Space Encoding

ASPC[1:0]	Type of Access
10	Program and Data
11	Program

D.9.2 BEFLASH Test Register

BFETST — BEFLASH Test Register **$YFF7A2**

Used during factory test only.

D.9.3 BEFLASH Base Address Registers

BFEBAH — BEFLASH Base Address High Register **$YFF7A4**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								ADDR 23	ADDR 22	ADDR 21	ADDR 20	ADDR 19	ADDR 18	ADDR 17	ADDR 16
RESET:															
								SB	SB	SB	SB	SB	SB	SB	SB

BFEBAL — BEFLASH Base Address Low Register **$YFF7A6**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR 15	ADDR 14	ADDR 13	ADDR 12	ADDR 11	0	0	0	0	0	0	0	0	0	0	0
RESET:															
SB	SB	SB	SB	SB	0	0	0	0	0	0	0	0	0	0	0

BFEBAH and BFEBAL contain the 13 high-order bits of the BEFLASH array base address. During reset, BFEBAH and BFEBAL take on the default values programmed into the associated shadow registers. After reset, if LOCK = 0 and STOP = 1, software can write to BFEBAH and BFEBAL to relocate the BEFLASH array. Because the states of ADDR[23:20] follow the state of ADDR19, addresses in the range $080000 to $F7FFFF cannot be accessed by the CPU16. If the BEFLASH array is mapped to these addresses, the system must be reset before the array can be accessed.

D.9.4 BEFLASH Control Register

BFECTL — BEFLASH Control Register **$YFF7A8**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	VFPE	ERAS	LAT	ENPE
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BFECTL contains the bits needed to control programming and erasing the BEFLASH.

VFPE — Verify Program/Erase

0 = Normal read cycles.

1 = Invoke program-verify circuit.

This bit invokes a special program-verify circuit. During programming sequences (ERAS = 0), VFPE is used in conjunction with the LAT bit to determine when programming of a location is complete. If VFPE and LAT are both set, a bit-wise exclusive-OR of the latched data with the data in the location being programmed occurs when any valid BEFLASH location is read. If the location is completely programmed, a value of zero is read. Any other value indicates that the location is not fully programmed. When VFPE is cleared, normal reads of valid BEFLASH locations occur.

ERAS — Erase Control

0 = BEFLASH configured for programming.

1 = BEFLASH configured for erasure.

The ERAS bit in BFECTL configures the BEFLASH array for programming or erasure. Setting ERAS causes all locations in the array and all BEFLASH shadow bits in the control block to be configured for erasure. **Table D-84** shows the address ranges that must be written to during an erase operation in order to erase specific blocks of the BEFLASH array.

Table D-84 BEFLASH Erase Operation Address Ranges

Block	Addresses Affected	Address Bits Used to Specify Block for Erasure						
		ADDR[23:11]	ADDR[10:6]	A5	A4	A3	A2	ADDR[1:0]
0	$0000 - $007F	BFEBAH/ BFEBAL[1]	X[2]	1	0	0	0	X[2]
1	$0080 - $0100			1	0	0	1	
2	$0100 - $017F			1	0	1	0	
3	$0180 - $01FF			1	0	1	1	
4	$0200 - $02FF			1	1	0	0	
5	$0300 - $03FF			1	1	0	1	
6	$0400 - $05FF			1	1	1	0	
7	$0600 - $07FF			1	1	1	1	
Entire Array[3]	$0000 - $07FF			0	X	X	X	

NOTES:
1. The block erasable flash base address high and low registers (BFEBAH and BFEBAL) specify ADDR[23:11] of the block to be erased.
2. These address bits are “don’t cares” when specifying the block to be erased.
3. Erasing the entire array also erases the BEFLASH control register shadow bits.

When the LAT bit is set, ERAS also determines whether a read returns the value of the addressed location (ERAS = 1) or the location being programmed (ERAS = 0).

The value of ERAS cannot be changed if the program/erase voltage is turned on (ENPE = 1).

LAT — Latch Control

0 = Programming latches disabled
1 = Programming latches enabled

When LAT is cleared, the BEFLASH address and data buses are connected to the IMB address and data buses. The BEFLASH is configured for normal reads. When LAT is set, the BEFLASH address and data buses are connected to parallel internal latches. The BEFLASH array is configured for programming or erasing.

Once LAT is set, the next write to a valid BEFLASH address causes the programming circuitry to latch both address and data. Unless control register shadow bits are to be programmed, the write must be to an array address.

The value of LAT cannot be changed when program/erase voltage is turned on (ENPE = 1).

ENPE — Enable Program/Erase

0 = Disable program/erase voltage
1 = Apply program/erase voltage

ENPE can be set only after LAT has been set, and a write to the data and address latches has occurred. ENPE remains cleared if these conditions are not met. While ENPE is set, the LAT, VFPE, and ERAS bits cannot be changed, and attempts to read a BEFLASH array location in BEFLASH are ignored.

D.9.5 BEFLASH Bootstrap Words

BFEBS[3:0] — BEFLASH Bootstrap Words **$YFF7B0 – $YFF7B6**

15															0
BOOTSTRAP VECTOR															
RESET:															
PROGRAMMED VALUE															

The BEFLASH bootstrap words (BFEBS[3:0]) can be used as system bootstrap vectors. When BOOT = 0 in BFEMCR during reset, the BEFLASH responds to program space accesses of IMB addresses $000000 to $000006 after reset. When BOOT = 1, the BEFLASH responds only to normal array and register accesses. BFEBS[3:0] can be read at any time, but the values in the words can only be changed by programming the appropriate locations.

INDEX

–A–

ABD 5-55, 5-59, 5-60
Accumulator
  M overflow flag (MV) 4-4, D-3
  offset addressing mode 4-10
ADC 10-1
  address map D-33
  analog subsystem 10-4
  block diagram 10-2
  clock 10-6
  conversion
    control logic 10-7
      modes 10-8
      parameters 10-7
    timing 10-12
  digital control subsystem 10-6
  external connections 10-1
  features 3-2
  overview 10-1
  prescaler 10-6
  programmer's model 10-3
  registers
    control registers (ADCTL) 10-6, D-35, D-36
    left justified
      signed (LJSRR) D-40
      unsigned (LJURR) D-41
    module configuration register (ADCMCR) D-34
    port ADA data register (PORTADA) D-34
    result registers 10-13
    right justified, unsigned (RJURR) D-40
    status register (ADSTAT) 10-6, D-40
    test register (ADTEST) D-34
ADCMCR 10-1, D-34
ADCTL D-35, D-36
ADCTST 10-1
ADDD 4-9
ADDE 4-9
ADDR
  bus signals 5-30
  definition 2-6
  signal 5-35
  starting address D-20
Address
  and data bus I/O availability 5-54
  bus (ADDR) 5-30
  bus disable (ABD) 5-55
  extension 4-6
    fields 4-5
    register 4-5
  map
    MC68HC16R1 3-22
    MC68HC16R3 3-24
    MC68HC916R1 3-23
    MC68HC916R3 3-25
  -mark wakeup 11-22
  space
    encoding 5-32
    maps 3-25
  strobe ($\overline{AS}$). *See* $\overline{AS}$ 5-30
Addressing modes 4-8
  accumulator offset 4-10
  extended 4-10
  immediate 4-9
  indexed 4-10
    use in replacing M68HC11 direct mode 4-11
  inherent 4-10
  post-modified index 4-10
  relative 4-10
ADSTAT D-40
ADTEST D-34
AIS 4-9
AIX/Y/Z 4-9
Alternate bus cycle termination 5-43
Analog
  input
    circuitry 10-15
    considerations 10-20
    pins 10-2, 10-22
      electrical model 10-22
  power pins 10-14
  reference pins 10-3, 10-14
  subsystem 10-4
  supply
    filtering and grounding 10-16
    pins 10-3
  -to-digital converter (ADC). *See* ADC 10-1
Arbitration priorities 5-64
$\overline{AS}$ 4-41, 5-40, 5-43, 5-45, 5-47
ASPC 7-2, 7-3, 8-3, 9-2, 9-3, D-29, D-87
Asserted (definition) 2-6
Asynchronous
  bus cycles classified as 5-37
  chip-select assertion 5-71
  cycles 5-30, 5-44
  exceptions 4-39, 5-48
  mode 5-72, 5-73
  resets 5-48, 5-49
Autocorrelation 4-46
Autovector ($\overline{AVEC}$). *See* $\overline{AVEC}$ 5-25
$\overline{AVEC}$ 5-25, 5-33, 5-43, 5-65, 5-66, 5-68, 5-70, 5-75
  enable bit 5-73, D-23

–B–

Background debug mode 4-40, 4-42, 5-41
- commands 4-43
- connector pinout 4-45
- enabling 4-42
- entering 4-43
- recommended connection 4-45
- serial
  - I/O block diagram 4-45
  - interface 4-44
- sources 4-42

Basic operand size 5-35
Baud
- clock 11-18
- rate 11-18

BCD 4-6
BEFLASH 5-60
- address map D-86
- array
  - address
    - mapping 9-2
    - space type 9-2
  - space field D-87
- erase operation address ranges 9-6, D-89
- erasure
  - flowchart 9-8
  - sequence 9-6
- features 3-2
- low-power stop mode operation 9-3
- normal access 9-3
- program/erase
  - operation 9-3
  - voltage signal conditioning 9-9
- programming
  - flowchart 9-5
  - sequence 9-4
- register block 9-2
- registers
  - array base address register
    - high (BFEBAH) 9-2, D-88
    - low (BFEBAL) 9-2, D-88
  - bootstrap words (BFEBS) 9-2, D-90
  - control register (BFECTL) 9-2, D-88
  - module configuration register (BFEMCR) 9-2, D-86
  - test register (BFETST) 9-2, D-87
- reset 9-3
- sectoring 9-1

$\overline{\text{BERR}}$ 5-33, 5-37, 5-41, 5-43, 5-44, 5-45, 5-54, 5-57, 5-62, 5-65, 5-66, 5-77, 5-78, 5-79
BFEBAH 9-2, D-88
BFEBAL 9-2, D-88
BFEBS 9-2, 9-3, D-90
BFECTL 9-2, D-88
BFEMCR 9-2, D-86
BFETST 9-2, D-87
$\overline{\text{BG}}$ 5-46, 5-58, 5-70
$\overline{\text{BGACK}}$ 5-46, 5-58, 5-70
Binary
- coded decimal (BCD) 4-6
- -weighted capacitors 10-5

Bit-time 11-17
BIUMCR D-57
BIUSM
- FREEZE D-57
- interrupt vector base number (VECT) D-57
- LPSTOP 12-6
- registers 12-6
  - module configuration register (BIUMCR) D-57
  - test configuration register (BIUTEST) D-58
  - time base register (BIUTBR) D-58
- STOP 12-5, D-57

BIUTBR D-58
BIUTEST D-58
$\overline{\text{BKPT}}$ 4-42, 5-41, 5-49, 5-61
Block
- -erasable flash EEPROM (BEFLASH). *See* BEFLASH 3-2
- size (BLKSZ) 5-70, D-20
  - encoding 5-71, D-20

BME 5-25, D-14
BMT 5-25, D-14
$\overline{\text{BOOT}}$ 7-3, 8-3, 9-3, D-28
- bit (BEFLASH) D-87
- bit (FLASH) D-81

Boot ROM control ($\overline{\text{BOOT}}$) D-28
Bootstrap words (ROMBS) 7-1
$\overline{\text{BR}}$ 5-46, 5-58, 5-69, 5-70
Break frame 11-17
Breakpoint
- acknowledge cycle 5-41
- exceptions 4-40
- hardware breakpoints 5-41
- mode selection 5-61
- operation 5-42

Breakpoints 4-41
BSL D-65
Buffer amplifier 10-5
Built-in emulation memory C-1
Bus
- arbitration for a single device 5-46
- cycle
  - alternate termination 5-43
  - regular 5-37
- error
  - exception processing 5-44
  - signal ($\overline{\text{BERR}}$). *See* $\overline{\text{BERR}}$ 5-25
  - timing of 5-44
- grant ($\overline{\text{BG}}$). *See* $\overline{\text{BG}}$ 5-46
- grant acknowledge ($\overline{\text{BGACK}}$). *See* $\overline{\text{BGACK}}$ 5-46
- interface unit submodule. *See* BIUSM 12-1
- monitor 5-25
  - external enable (BME) D-14
  - timeout period 5-25, D-15
  - timing (BMT) 5-25, D-14
- request ($\overline{\text{BR}}$). *See* $\overline{\text{BR}}$ 5-46
- select (BSL) D-65
- state analyzer 4-40

BYTE (upper/lower byte option) 5-72, D-21

–C–

C 4-4, D-3
Carry flag (C) 4-4, D-3
Case outline (831A) B-6
CCF D-40
CCR 4-4, 5-64, D-3
CCTR D-40
CD/CA D-37
C_{DAC} 10-23
Central processing unit (CPU16). *See* CPU16 4-1
C_F 10-23
Channel selection for A/D conversion D-37
Charge sharing 10-24
Chip-select
  address compare logic 5-71
  assertion 5-66
  base address registers (CSBAR) 5-69, 5-70
    reset values 5-75
  block (SCIM2) 5-1
  circuit block diagram 5-68
  operation 5-73
  option registers (CSOR) 5-69, 5-71, D-21
    reset values 5-75
  pin assignment registers (CSPAR) 5-21, 5-68, 5-69, D-19
    field encoding 5-70, D-18
  registers 5-68
  reset operation 5-75
  signals for interrupt acknowledge cycle termination 5-74
Clear (definition) 2-6
CL_I 4-37
Clipping errors 10-16
CLK D-61, D-63, D-76
CLKOUT 5-37, 5-48
CL_O 4-37
Clock
  ADC 10-6
  frequency (calculation) D-7
  input pin status (FCSM) D-63
  input pin status (MCSM) D-60
  output (CLKOUT) 5-37
  phase (CPHA) 11-8
    = 0 transfer format 11-9
    = 1 transfer format 11-10
  polarity (CPOL) 11-8
  rate selection (CLK) field D-76
  synthesizer 5-5
Closed-loop control routines 4-46
CL_P 4-37
CL_T 4-37
COF D-60, D-62
Combined program and data space maps
  MC68HC16R1 3-27
  MC68HC16R3 3-31
  MC68HC916R1 3-29
  MC68HC916R3 3-33
Comparator 10-6
Condition code register (CCR) 4-4
Conventions 2-6
Conversion
  complete flags (CCF) D-40
  control logic 10-7
    modes 10-8
    parameters 10-7
  counter (CCTR) D-40
  timing 10-12
Counter
  clock select (CLK) field
    FCSM D-63
    MCSM D-61
  overflow flag (COF) bit D-60, D-62
CPCR D-58
CPHA 11-8, 11-9, 11-10
CPOL 11-8
CPSM
  block diagram 12-7
  registers 12-7
    control register (CPCR) D-58
    test register (CPTR) D-59
CPTR D-59
CPU space 5-74
  address encoding 5-41
  cycles 5-40, 5-74
  encoding for interrupt acknowledge 5-74
  read cycle 5-64
CPU16 4-1, 5-21, 5-25, 5-29, 5-32, 5-35, 5-36, 5-41, 5-42, 5-45, 5-48, 5-61, 5-63, 5-66, 5-71, 5-76
  accumulators 4-3
  address
    extension 4-6
    extension register 4-5
  addressing modes 4-8
    accumulator offset 4-10
    extended 4-10
    immediate 4-9
    indexed 4-10
    inherent 4-10
    post-modified index 4-10
    relative 4-10
  breakpoints 4-41
  compatibility with M68HC11 4-1
  condition code register (CCR) 4-4
  data types 4-6
  exception 5-48
  extension fields 4-6
  features 3-1
  general information 4-1
  index registers 4-3
  instruction 4-11
    comparison to M68HC11 4-31
    execution model 4-34
    set
      abbreviations and symbols 4-30
      summary 4-12
    timing 4-36
  memory
    management 4-5
    organization 4-7

program counter (PC) 4-3
reference manual 4-1
register model 4-2, D-2
registers
condition code register (CCR) 5-63, D-3
mnemonics 2-2
multiply and accumulate (MAC) registers 4-5
stack pointer (SP) 4-3
CPU32 5-59, 5-76
CPUD 5-58, 5-59, 5-60
CREG D-24
Cross-correlation 4-46
Crystal oscillator 5-6
CSBAR D-20
CSBARBT 5-71, D-20
$\overline{\text{CSBOOT}}$ 5-49, 5-58, 5-66, 5-68, 5-69, 5-70, 5-71, 5-75, 5-76, 7-3
reset values 5-76
$\overline{\text{CSE}}$ 5-58, 5-61, 5-62, 5-77
$\overline{\text{CSM}}$ 5-58, 5-59, 5-61
CSOR D-21
CSORBT D-21
CSPAR 5-21, 5-58
CSPAR0/1 D-18
CTD9 D-60
CTM
address map D-55
components 12-1
CTM Reference Manual 12-1
CTM2C D-60
CTM7
features 3-2

–D–

DAC capacitor array (C_{DAC}) 10-23
DASM
block diagram 12-15
channels 12-14
interrupts 12-13, 12-16
mode flag status bit states D-65
modes of operation 12-14
registers 12-14, 12-16
data register A (DASMA) D-67
data register B (DASMB) D-68
DASMA D-67
operations D-68
DASMB D-68
operations D-69
DATA 5-30
Data
and size acknowledge ($\overline{\text{DSACK}}$). *See* $\overline{\text{DSACK}}$ 5-25
bus
mode select conditioning circuits
alternate 5-57
preferred 5-56
mode selection 5-55
signals (DATA) 5-30
frame 11-17
multiplexer 5-35
strobe ($\overline{\text{DS}}$). *See* $\overline{DS}$ 5-32
DATA (definition) 2-6
$\overline{\text{DATA14}}$ 8-3
$\overline{\text{DATA15}}$ 9-3
DDRAB 5-57, D-11
DDRE D-11
DDRF D-12
DDRG D-10
DDRH D-10
DDRM 11-2, D-46
Designated CPU space 5-32
Development
support for CPU16 4-40
tools and support C-1
Digital
control subsystem 10-6
signal processing (DSP) 4-46
DIS D-68, D-69
Divide
by 2/divide by 3 (DIV23) D-59
-by-128 circuit 5-5
Double
-buffered 11-19, 11-20
bus fault 5-45
DREG D-24
Drive time base bus (DRV) D-60, D-63
Driver types 3-15
DRV D-60, D-63
$\overline{\text{DS}}$ 4-41, 5-45, 5-47, 5-55
$\overline{\text{DSACK}}$ 5-25, 5-30, 5-33, 5-37, 5-41, 5-43, 5-44, 5-45, 5-65, 5-66, 5-68, 5-70, 5-71, 5-73
external/internal generation 5-40
option fields 5-40
signal effects 5-34
source specification in asynchronous mode 5-72, D-22
DSCLK 4-44
DSP 4-46
Dynamic bus sizing 5-33

–E–

ECLK 5-21, 5-58
Edge polarity (EDPOL) bit D-66
EDGEN D-60
EDGEP D-60
EDIV 5-21, D-8
EDPOL D-66
EK 4-5
Electrical characteristics A-1
EMUL D-29
Emulation mode
control (EMUL) D-29
selection 5-62
Emulator chip select ($\overline{\text{CSE}}$) 5-62
EN D-75
Enable program/erase (ENPE) D-84, D-89
ENPE D-84, D-89
ERAS D-84, D-88
Erase control (ERAS) D-84, D-88

Error conditions 11-21
EV 4-4, D-3
Event flag (FLAG) D-64
Exception
asynchronous 4-39
definition 4-37
multiple 4-40
processing 5-48
sequence 4-39
stack frame 4-38
format 4-38
synchronous 4-39
types 4-39
vector 4-37, 5-48
table 4-38
Execution
process 4-36
unit 4-35
EXOFF D-6
EXT D-9
EXTAL 5-6, 5-8, 5-21, 5-51, 5-62
Extended addressing modes 4-10
Extension
bit overflow flag (EV) 4-4, D-3
field (SK) 4-3
fields 4-6
External
bus
arbitration 5-46
clock
division (EDIV) 5-21, D-8
signal (ECLK) 5-21
interface (EBI) 5-1, 5-30
bus control signals 5-30
circuit settling time 10-24
clock
mode 5-20
off (EXOFF) bit D-6
leakage 10-24
multiplexing of analog signal sources 10-21
reset (EXT) D-9

–F–

Factory test 5-81
Fast
reference mode 5-14
clock
control multipliers 5-17
frequencies 5-18
speed changes 5-17
termination cycles 5-36, 5-39
FASTREF 5-5, 5-8, 5-16, 5-21, 5-54
FC 5-32
FCSM
block diagram 12-8
clock sources 12-8
counter 12-8
external event counting 12-9
interrupts 12-9
registers 12-9
counter register (FCSMCNT) D-64
status/interrupt/control register (FCSMSIC) D-62
time base bus drivers 12-9
FCSMCNT D-64
FCSMSIC D-62
FE 11-21, D-51
FEExBAH 8-1, D-82
FEExBAL 8-1, D-82
FEExBS D-85
FEExCTL 8-1, D-83
FEExMCR 8-1, D-80
FEExTST 8-1, D-82
Ferrite beads 10-14
FLAG D-64, D-74
Flash EEPROM
address map D-79
array
space field D-81
array address
mapping 8-2
space type 8-2
bootstrap
words D-85
erasure
flowchart 8-7
sequence 8-6
features 3-2
low-power stop mode 8-3
normal access 8-2
not supported in emulation mode 5-62
program/erase
operation 8-3
voltage signal conditioning 8-8
programming
flowchart 8-5
sequence 8-4
register block 8-1
registers
array base address register
high (FEExBAH) 8-1, D-82
low (FEExBAL) 8-1, D-82
bootstrap words (FEExBS) D-85
control register (FEExCTL) 8-1, D-83
module configuration register (FEExMCR) 8-1, D-80
test register (FEExTST) 8-1, D-82
reset 8-3
FORCA D-66
FORCB D-66
Force (FORCA/B) D-66
f_{PWM} 12-20
Frame 11-17
size 11-21
Framing error (FE) flag 11-21, D-51
FREEZE 4-43
assertion response (FRZ)
ADC 10-3, D-34
BIUSM 12-5, D-57

SCIM2 5-4
bus monitor (FRZBM) D-6
mode control (FRZ) D-81, D-87
software enable (FRZSW) D-6
f_{ref} 5-5, 5-14, 5-17, 5-23, 5-51
Frequency
control bits
counter (Y) D-8
prescaler (X) D-8
VCO (W) D-8
dividers 5-6
FRZ 10-3, D-34, D-81, D-87
FRZBM D-6
FRZSW D-6
f_{sys} 5-5, 5-10, 5-12, 5-17, 5-19, 12-20, D-7
F-term encoding 5-40
Fully (16-bit) expanded mode 5-59
Function code (FC) signals 5-32, 5-40
f_{VCO} 5-10, 5-12, 5-17

–G–

Gain 10-19
General-purpose I/O ports 5-76
Grounding 10-17

–H–

H 4-4, D-3
Half carry flag (H) 4-4, D-3
$\overline{\text{HALT}}$ 5-25, 5-33, 5-37, 5-43, 5-44, 5-45, 5-46
Halt
monitor 5-25
enable (HME) 5-25, D-14
reset (HLT) D-9
operation 5-45
negating/reasserting 5-45
Handshaking 5-36
Hardware breakpoints 5-41
HCMOS 1-1
High-density complementary metal-oxide semiconductor (HCMOS) 1-1
HLT D-9
HME 5-25, D-14

–I–

IACK 5-64
IARB
BIUSM D-58
MCCI D-43
SCIM2 5-3, D-7
IARB3 D-60, D-63, D-65, D-74
IC D-68, D-69
ICD16/ICD32 C-1
IDLE 11-21, D-51
Idle
frame 11-17
-line
detect type (ILT) 11-21, D-49
detected (IDLE) 11-21, D-51
detection process 11-21
interrupt enable (ILIE) 11-22, D-49
I_{FPE} 8-3, 9-4
I_{IN} 10-19
IL D-60, D-63, D-65, D-74
ILIE 11-22, D-49
ILSCI 11-2, D-43
ILSCIA D-43
ILSCIB D-43
ILSPI 11-2, D-44, D-45
ILT 11-21, D-49
IMB 12-1
IMM16 4-9
IMM8 4-9
Immediate addressing modes 4-9
Impedance 10-22
IN D-63, D-65
IN1 D-60
IN2 D-60
In-circuit debugger (ICD16/ICD32) C-1
Indexed addressing modes 4-10
Inductors 10-14
Inherent addressing modes 4-10
Input
leakage errors 10-24
pin status (IN)
DASM D-65
Instruction
execution model 4-35
fetches 4-7
pipeline 4-35
set for CPU16 4-11
timing 4-36
Intermodule bus (IMB) 3-3, 12-1
Internal
bus
error ($\overline{\text{BERR}}$) 5-25
monitor 5-25
clock signals (PCLK) D-61
register map 3-21
Interrupt
acknowledge and arbitration 5-64
arbitration 5-3
IARB field
BIUSM D-58
MCCI D-43
SCIM2 D-7
IARB3 bit
DASM D-65
FCSM D-63
MCSM D-60
PWMSM D-74
exception processing 5-63
level (IL)
DASM D-65
FCSM D-63
MCSM D-60
PWMSM D-74

priority
and recognition 5-63
level field (IPL) D-23
mask (IP) field 4-4, 5-63, D-3
processing summary 5-66
vector D-44
Interrupts
DASM 12-13, 12-16
FCSM 12-9
MCSM 12-12
SCIM2 5-63
INTV D-44
I_{OUT} 10-19
IP 4-4, D-3
IPIPE0 4-43, 5-58, 5-59, 5-60
IPIPE1 4-43, 5-58, 5-59, 5-60
IPL D-23
IPM D-68, D-69
IPWM D-68, D-69
$\overline{IRQ7}$ 5-63, 5-64
I_{SB} 6-3
IX 4-3
IY 4-3
IZ 4-3

–J–

Junction leakage 10-24

–L–

LAT D-84, D-89
Latch control (LAT) D-84, D-89
Leakage error 10-24
Limp status (SLIMP) 5-24
LJSRR D-40
LJURR D-41
LOAD D-74
LOC D-9
LOCK 7-3, 9-3, D-28, D-81, D-87
Lock registers (LOCK) D-28, D-81, D-87
Logic
analyzer pod connectors C-2
levels (definition) 2-6
Loop mode (LOOPS) D-48
LOOPS D-48
LOSCD 5-23, 5-24
Loss
of clock reset (LOC) D-9
of reference signal 5-23
-of-clock oscillator (LOSCD) 5-24
Low-power
broadcast cycle 5-42
CPU space cycle 5-42
interrupt mask level 5-42
operation 5-21
stop mode enable (STOP)
ADC 10-3, D-34
BEFLASH D-86
BIUSM D-57
Flash EEPROM D-81
MCCI 11-2, D-43
MRM 7-3, D-28
SCIM2 5-29
SRAM 6-2, D-25
LPSTOP 5-21, 5-29, 5-49
effect on BIUSM 12-6
flowchart 5-23
LSB 2-6
LSBF 11-11
LSW 2-6

–M–

M 11-18, D-49
M6800 bus E clock signal 5-58, 5-71
M68HC11 instructions compared to CPU16 instructions 4-31
M68MEVB1632 modular evaluation board (MEVB) C-1
M68MMDS1632 modular development system (MMDS) C-1
MAC 4-5, 4-9, 4-46
Masked ROM module (MRM). *See* MRM 7-1
MCCI
address map 11-2, D-42
block diagram 11-1
features 3-2
general 11-1
general-purpose I/O 11-4
initialization 11-23
interrupts 11-3
pin function D-46
reference manual 11-1
registers
data direction register (MDDR) 11-4
global registers
data direction register (DDRM) 11-2, D-46
interrupt vector register (MIVR) 11-2, D-44
module configuration register (MMCR) 11-2, D-42
pin control registers
pin assignment register (MPAR) 11-2, D-45
port
data register (PORTMC) 11-2, D-47
pin state register (PORTMCP) 11-2, D-47
SCI
interrupt level register (ILSCI) 11-2, D-43
SPI
interrupt level register (ILSPI) 11-2, D-44
test register (MTEST) 11-2, D-43
pin assignment register (MPAR) 11-4
SCI
control register 0 (SCCR0A/B) 11-13, D-47
control register 1 (SCCR1A/B) 11-16, D-48
data register (SCDRA/B) 11-16, D-51

status register (SCSRA/B) 11-16, D-50
SPI
control register (SPCR) 11-6, D-52
data register (SPDR) 11-6, D-54
status register (SPSR) 11-6, D-53
types 11-2
SCI 11-13
interrupt level (ILSCIA, ILSCIB) D-43
SPI 11-4
MCSM
block diagram 12-10
clock sources 12-11
counter 12-10
external event counting 12-11
interrupts 12-12
modulus latch 12-10
registers 12-12
counter register (MCSMCNT) D-61
modulus latch register (MCSMML) D-62
status/interrupt/control register (MCSMSIC) D-59
time base bus drivers 12-11
MCSMCNT D-61
MCSMML D-62
MCSMSIC D-59
MCU
132-pin assignment package
HC16R1 3-8, B-2
HC16R3 3-10, B-4
HC916R1 3-9, B-3
HC916R3 3-11, B-5
basic system 5-31
block diagram
HC16R1 3-4
HC16R3 3-6
HC916R1 3-5
HC916R3 3-7
components 1-1
overview 1-1
personality board (MPB) C-1
MDDR 11-4
Mechanical data and ordering information B-1
Memory maps
combined program and data
MC68HC16R1 3-27
MC68HC16R3 3-31
MC68HC916R1 3-29
MC68HC916R3 3-33
internal register map 3-21
separate program and data
MC68HC16R1 3-28
MC68HC16R3 3-32
MC68HC916R1 3-30
MC68HC916R3 3-34
Microcontroller Development Tools Directory (MCUDEVTLDIR/D Rev. 3) C-1
Microsequencer 4-35
Misaligned operand 5-35
MIVR 11-2, D-44
MM 5-2, 11-2, 12-23, 12-24, D-1, D-5, D-6, D-7, D-25, D-27, D-33, D-42, D-57, D-80, D-86
MMCR 11-2, D-42
MMDS C-1
Mnemonics
range (definition) 2-6
specific (definition) 2-6
MODE 5-71, D-21, D-66
Mode
fault flag (MODF) 11-12, D-53
select (M) D-49
MODF D-53
Modular platform board C-1
Module mapping (MM) bit. *See* MM 3-21
Modulus
counter 11-18
load
edge sensitivity (EDGEN, EDGEP) bits D-60
input pin status (IN1) D-60
Monotonicity 10-1
MOSFET 5-57
MPAR 11-2, 11-4, D-45
MPB C-1
MRM 7-1
address map D-27
array address mapping 7-1
features 3-2
low-power stop operation 7-3
normal access 7-2
registers
module configuration register (MRMCR) 7-1, D-28
ROM
array base address registers (ROMBAH/BAL) 7-1, D-29
bootstrap words (ROMBS) 7-1, D-31
signature registers (RSIGHI/LO) 7-1, D-31
reset 7-3
ROM signature 7-3
MRMCR 7-1, D-28
MSB 2-6
MSTR 11-7, 11-8
MSW 2-6
MTEST 11-2, D-43
MULT D-36
Multichannel
communication interface module (MCCI). *See* MCCI 11-1
conversion (MULT) D-36
Multiple
-channel conversion 10-11, D-39
exceptions 4-40
Multiplexer 10-4
channels 10-4
Multiply and accumulate (MAC) 4-46
MV 4-4, D-3

–N–

N 4-4, D-3
N_{CLOCK} 12-20

Negated (definition) 2-6
Negative
flag (N) 4-4, D-3
integers 4-6
stress 10-18
NF 11-21, D-51
Noise 10-14
error (NF) D-51
errors 11-21
flag (NF) 11-21
reduction 5-3
Non-maskable interrupt 5-64
N_{PERIOD} 12-20
NRZ 11-2, 11-13

–O–

OCAB D-68, D-69
OCB D-68, D-69
OP (1 through 3) 5-34
Opcode tracking 4-40
breakpoints 4-42
combining with other capabilities 4-41
deterministic 4-40
Operand
alignment 5-35
byte order 5-34
misaligned 5-35
transfer cases 5-35
Operating
configuration out of reset 5-53
mode selection 5-54
Operators 2-1
OPWM D-68, D-69
OR D-51
Output
flip-flop 12-17
pin
polarity control (POL) bit D-75
status (PIN) bit D-74
Overflow flag (V) 4-4, D-3
Overrun error (OR) D-51
Overview information 3-1

–P–

Parasitic devices 10-18
Parentheses (definition) 2-6
Parity
checking 11-19
enable (PE) D-49
error (PF) flag 11-21, D-51
errors 11-21
type (PT) 11-19, D-49
Partially (8-bit) expanded mode 5-60
PC 4-3
PCLK D-61
PCLK6 D-59
PE D-49
PEPAR D-11
Period
and pulse width register load control (LOAD) bit D-74
completion status (FLAG) bit D-74
Periodic
interrupt
modulus counter 5-28
priority 5-29
request level (PIRQL) D-15
timer 5-28
components 5-28
interrupt priority and vectoring 5-29
modulus (PITM field) 5-28, D-15
PIT period calculation 5-28, D-16
vector (PIV) D-15
timer prescaler control (PTP) 5-28, D-15
PF 11-21
PFPAR D-12
Phase-locked loop (PLL). *See* PLL 1-1
PICR D-15
PIN D-74
Pin
associations with basic configuration options 5-54
considerations 10-14
reset states 5-52
Pipeline multiplexing 4-41
PIRQL D-15
PITM 5-28, D-15
PITR 5-28, D-15
PIV D-15
PK 4-4, 4-5, D-3
PLL 1-1, 5-1, 5-5, 5-51
POL D-75
POR 5-50
Port
A 5-77
data register (PORTA) 5-77
B 5-77
data register (PORTB) 5-77
C
data register (PORTC) 5-73
E 5-77
data direction register (DDRE) 5-78
data register (PORTE) 5-77
pin assignment register (PEPAR) 5-78
F 5-78
block diagram 5-79
data direction register (DDRF) 5-79
data registers (PORTF1, PORTF0) 5-79
edge-detect flag register (PORTFE) 5-80
edge-detect interrupt level register (PFLVR) 5-78, 5-80
edge-detect interrupt vector register (PFIVR) 5-78, 5-80
pin assignment register (PFPAR) 5-79
G 5-80
data direction register (DDRG) 5-80
data register (PORTG) 5-80
H 5-80
data direction register (DDRH) 5-81

data register (PORTH) 5-80
replacement unit (PRU) C-2
size 5-70
PORTA D-9
PORTADA 10-1, D-34
PORTB D-9
PORTC 5-68, D-17
PORTE D-10
PORTF0/1 D-12
PORTG D-10
PORTH D-10
PORTMC 11-2, D-47
PORTMCP 11-2, D-47
Positive stress 10-18
Post-modified index addressing mode 4-10
POW D-9
Power
-on reset (POR) 5-50
-up reset (POW) D-9
Prescaler
division ratio select (PSEL) D-59
rate selection field (PRS) D-35
running (PRUN) D-58
Program
counter address extension field (PK) 4-4, D-3
flow changes 4-36
Programming voltage envelope 8-8, 9-9
PRS D-35
PRU C-2
PRUN D-58
PSEL D-59
PSHM 4-10
PT 11-19, D-49
PTP D-15
PULM 4-10
PWMA D-76
PWMB D-77
PWMC D-77
PWMSIC D-73
PWMSM
block diagram 12-17
clock selection 12-17
coherency 12-19
counter 12-18
enable (EN) D-75
output flip-flop 12-17
period registers and comparator 12-18
pulse width registers and comparator 12-19
PWM
duty cycle boundary cases 12-22
frequency 12-20
period and pulse width register values 12-21
pulse width 12-21
registers 12-22
PWM counter register (PWMC) D-77
PWM period register (PWMA) D-76
PWM pulse width register (PWMB) D-77
PWM status/interrupt/control register (PWMSIC) D-73

–Q–

QUOT 5-81

–R–

R/$\overline{W}$ 5-3, 5-4, 5-32, 5-37, 5-40, 5-55, 5-58
field 5-72, D-21
RAF D-50, D-51
RAM
array space (RASP) D-25
base address lock (RLCK) bit D-25
RAMBAH 6-1, D-26
RAMBAL 6-1, D-26
RAMMCR 6-1, D-25
RAMTST 6-1, D-26
RASP 6-2, D-25
encoding 6-2, D-26
RC
DAC array 10-5
low pass filter 10-16
oscillator 5-23
RDRF 11-21, D-50
RE 11-4, 11-13, 11-20, D-50
Read
/write signal (R/$\overline{W}$). *See* R/$\overline{W}$ 5-32
cycle 5-38
flowchart 5-38
Receive
data
(RXDA/B) pins 11-17
register full (RDRF) D-50
time sample clock (RT) 11-18, 11-21
Receiver
active (RAF) D-50, D-51
data register (RDRF) flag 11-21
enable (RE) 11-4, 11-13, 11-20, D-50
interrupt enable (RIE) D-49
wakeup (RWU) 11-22, D-50
Register bit and field mnemonics 2-3
Relative addressing modes 4-10
$\overline{RES10}$ 10-7, D-35
$\overline{RESET}$ 5-1, 5-5, 5-8, 5-48, 5-49, 5-50, 5-51, 5-52, 5-54, 5-55, 5-57, 5-60, 5-62, 5-77
Reset
configuration for M68HC16 R-series memory modules 5-60
enable (RSTEN) 5-24
exception processing 5-48
operation in SCIM2 5-48
processing summary 5-49
source summary 5-48
states of pins assigned to other MCU modules 5-53
status register (RSR) 5-24, 5-25, 5-50
timing 5-50
Resistor-divider chain 10-5
Resolution 10-7
bit (RES10) D-35
Result registers 10-13

format 10-14
Return-from-interrupt instruction (RTI) 4-40
R_F 10-23
RF energy 10-14
RIE D-49
RJURR D-40
RLCK 6-1, D-25
RMAC 4-9
$\overline{\text{RMC}}$ 5-59
ROM array space (ASPC) D-29
ROMBAH 7-1, D-29
ROMBAL 7-1, D-29
ROMBS 7-1
ROMBS0-3 D-31
RS-232C terminal C-2
RSIGHI 7-1, 7-3, D-31
RSIGLO 7-1, 7-3, D-31
RSR 5-24, 5-25, D-9
RSTEN 5-24
RT 11-21
RTI 4-40
RWD 5-58, 5-59, 5-60
RWU 11-22, D-50
RXDA/B 11-16, 11-17

–S–

S 4-4, D-3
S8CM D-36
Sample
capacitor 10-5
time 10-7
time selection (STS) field D-35
SAR 10-13
Saturate mode (SM) bit 4-4, D-3
SBK 11-20, D-50
SCAN D-36
Scan mode selection bit (SCAN) D-36
SCBR D-47
SCCR0A 11-13, D-47
SCCR0B 11-13, D-47
SCCR1A/B 11-16, D-48
SCDRA/B 11-16, D-51
SCF D-40
Schottky diode 8-9, 9-10
SCI 11-1
baud
clock 11-18
rate (SCBR) 11-18, D-47
equation D-47
idle-line detection 11-21
internal loop 11-22
interrupt level (ILSCIA, ILSCIB) D-43
operation 11-17
parity checking 11-19
pins 11-16
receiver
operation 11-20
wakeup 11-22
receiver block diagram 11-15
registers
control register 0 (SCCR0A/B) 11-13, D-47
control register 1(SCCR1A/B) 11-16, D-48
data register (SCDRA/B) 11-16, D-51
status register (SCSRA/B) 11-16, D-50
serial formats 11-18
transmitter block diagram 11-14
SCIA 11-2
SCIB 11-2
SCIM Reference Manual (SCIMRM/AD) 5-1, 5-73
SCIM2 5-1
address map D-4
block diagram 5-2
bus operation 5-36
chip-select 5-66
clock signal (SCIMCLK) 5-22
enhancements 5-1
external bus interface (EBI) 5-30
features 3-1
general information 5-1
general-purpose I/O ports 5-76
halt monitor 5-25
interrupt arbitration 5-3
interrupts 5-63
operating modes 5-1
periodic interrupt timer 5-28
block diagram (with software watchdog) 5-27
pin reset states 5-52
registers
chip-select
base address
register boot (CSBARBT) D-20
registers (CSBAR) 5-69, 5-70, D-20
option
register boot (CSORBT) D-21
registers (CSOR) 5-69, 5-71, D-21
pin assignment registers (CSPAR) 5-69, D-18
clock synthesizer control register (SYNCR) D-7
distributed register (DREG) D-24
master shift registers A/B (TSTMSRA/TSTMSRB) D-23
module configuration register (SCIMCR) 5-2, D-5
periodic interrupt
control register (PICR) D-15
timer register (PITR) 5-28, D-15
port A
data register (PORTA) 5-77, D-9
port A/B data direction register (DDRAB) 5-77, D-11
port B
data register (PORTB) 5-77, D-9
port C data register (PORTC) 5-73, D-17
port E
data direction register (DDRE) 5-78, D-11
data register (PORTE) 5-77, D-10
pin assignment register (PEPAR) 5-78, D-11
port F

data direction register (DDRF) 5-79, D-12
data registers (PORTF1, PORTF0) 5-79, D-12
edge-detect flag register (PORTFE) 5-80
edge-detect interrupt level register (PFLVR) 5-78, 5-80
edge-detect interrupt vector register (PFIVR) 5-78, 5-80
pin assignment register (PFPAR) 5-79, D-12
port G
data direction register (DDRG) 5-80, D-10
data register (PORTG) 5-80, D-10
port H
data direction register (DDRH) 5-81, D-10
data register (PORTH) 5-80, D-10
reset status register (RSR) D-9
single-chip test register (SCIMTR) D-7
software service register (SWSR) D-16
system
protection control register (SYPCR) D-13
test register E (SCIM2TRE) D-9
test
module
repetition count register (TSTRC) D-24
shift count register (TSTSC) D-24
submodule control register (CREG) D-24
reset 5-48
software watchdog 5-26
spurious interrupt monitor 5-25
system
clock 5-4
protection 5-24
SCIM2TRE D-9
SCIMCLK 5-22
SCIMCR 5-2, D-5
SCIMTR D-7
SCSRA/B 11-16, D-50
Select eight-conversion sequence mode (S8CM) D-36
Send break (SBK) 11-20, D-50
Separate program and data space map
MC68HC16R1 3-28
MC68HC16R3 3-32
MC68HC916R1 3-30
MC68HC916R3 3-34
Sequence complete flag (SCF) D-40
Serial
clock
frequency range 4-44
communication interface (SCI). *See* SCI 11-13
data word 4-44
formats 11-18
interface clock signal (DSCLK) 4-44
mode (M) bit 11-18
peripheral interface (SPI). *See* SPI 11-4
shifter 11-19
Set (definition) 2-6
Settling time 10-23
SHEN 5-47, D-6
Show cycle
enable (SHEN) 5-4, 5-47, D-6
operation 5-47
Signature registers (RSIGHI/LO) 7-1
Signed fractions 4-6
Single
-channel conversions 10-10, D-38
-chip
integration module 2. *See* SCIM2 5-1
mode 5-57, 5-58
SIZ 5-32, 5-35, 5-47
SIZE (MCCI) 11-11
Size signals (SIZ) 5-32
encoding 5-32
SK 4-3, 4-5
SLIMP 5-24
SLOCK 5-6, 5-19, D-8
Slow reference mode 5-6
clock
control multipliers 5-10
frequencies 5-12
speed changes 5-9
SM 4-4, D-3
SMB 12-1
Software
service register (SWSR) 5-26
watchdog 5-26
block diagram 5-27
enable (SWE) 5-26, D-13
prescale (SWP) 5-26, D-13
ratio of SWP and SWT bits 5-27
reset (SW) D-9
timeout period calculation 5-26, D-14
timing field (SWT) 5-26, D-13
Source voltage level (V_{SRC}) 10-23
SP 4-3
SPACE (address space select) 5-72, D-22
SPCR 11-6, D-52
SPDR 11-6, D-54
SPI 11-1
block diagram 11-5
clock phase/polarity controls 11-8
finished flag (SPIF) D-53
interrupt level (ILSPI) D-45
mode fault 11-12
operating modes
master mode 11-7
slave mode 11-8
pins 11-6
registers
control register (SPCR) 11-6, D-52
data register (SPDR) 11-6, D-54
status register (SPSR) 11-6, D-53
serial clock baud rate 11-11
transfer 11-5
transfer size and direction 11-11
write collision 11-12
SPIF D-53
SPSR 11-6, D-53
Spurious interrupt exception 5-64

SRAM
address map D-25
array address mapping 6-1
features 3-1
normal access 6-2
registers
array base address register
high (RAMBAH) 6-1, D-26
low (RAMBAL) 6-1, D-26
module configuration register (RAMMCR) 6-1, D-25
test register (RAMTST) 6-1, D-26
reset 6-3
standby and low-power stop operation 6-2
Standard non-return to zero (NRZ) 11-2, 11-13
Star-point ground system 10-17
Start bit (beginning of data frame) 11-17
State machine 11-20
STEXT 5-22, D-8
STOP 5-29, 10-3, 11-2, D-25, D-28, D-34, D-43, D-57, D-81, D-86
enable (S) 4-4, D-3
Stop
enable (STOP) bit
BIUSM 12-5
mode
control D-86
external clock (STEXT) 5-22, D-8
SCIM clock (STSCIM) 5-22, D-8
mode control D-81
SCI end of data frame bit 11-17
STRB (address strobe/data strobe) bit 5-40, 5-72, D-22
Stress conditions 10-18
STS D-35
STSCIM 5-22, D-8
Submodule bus (SMB) 12-1
Successive approximation register (SAR) 10-13
Supervisor/unrestricted data space (SUPV). *See SUPV* D-6
SUPV 5-76, 11-3, D-34, D-43
SW D-9
SWE 5-26, D-13
SWP 5-26, D-13
SWSR 5-26, D-16
SWT 5-26, D-13
Symbols 2-1
Synchronization to CLKOUT 5-37
Synchronous
chip-select assertion 5-71
exceptions 4-39
mode 5-72
resets 5-48, 5-49
SYNCR 5-9, 5-14, 5-17, 5-19, D-7
Synthesizer lock (SLOCK) 5-6, D-8
SYPCR D-13
SYS D-9
System
clock 5-1, 5-4
output (CLKOUT) 5-37
sources 5-4
configuration 5-1, 5-2
protection block (SCIM2) 5-1
reset (SYS) D-9
test block (SCIM2) 5-1
test register E (SCIM2TRE) D-9

–T–

TBB 12-1
TBRS1 D-58
TBRS2 D-58
TC 11-19, D-50
TCIE 11-20, D-49
TDRE 11-19, D-50
TE 11-4, 11-13, D-49
Test submodule reset (TST) D-9
Three-state control (TSC) 5-62
TIE 11-20, D-49
Time base
bus driver for MCSM 12-11
buses (TBB) 12-1, 12-3, 12-4
allocation 12-5
register bus select bits (TBRS1/0) D-58
Timeout period calculations 5-26
t_{PWMAX} 12-21
t_{PWMIN} 12-21
Transmit
complete
flag (TC) 11-19, D-50
interrupt enable (TCIE) 11-20, D-49
data
pins (TXDA/B) 11-17
register empty (TDRE) flag 11-19, D-50
interrupt enable (TIE) 11-20, D-49
Transmitter enable (TE) 11-4, 11-13, 11-19, D-49
TSC 5-62
TST D-9
TSTMSRA D-23
TSTMSRB D-23
TSTRC D-24
TSTSC D-24
Two/three wire transfers 11-4
Two’s complement 4-6
TXDA/B 11-16, 11-17

–U–

UART 11-2, 11-13
Universal asynchronous receiver/transmitter (UART) 11-2, 11-13

–V–

V 4-4, D-3
V_{CF} 10-23
VCO 5-6, 5-10, 5-12
V_{DD} 5-50, 5-51, 6-1
V_{DDA} 10-1, 10-3, 10-14
V_{DDSYN}/MODCLK 5-5, 5-8, 5-16, 5-21, 5-51, 5-54

VECT D-57
Vector sources D-44
Verify program/erase (VFPE) D-84, D-88
VFPE D-84, D-88
V_{FPE} 8-3, 8-8, 9-1, 9-4, 9-9
    conditioning circuit 8-9, 9-10
V_I 10-23
V_{IH} 10-2
V_{IL} 10-2
Voltage
    controlled oscillator (VCO) 5-6
    limiting diodes 10-20
V_{PP} C-2
V_{RH} 10-1, 10-3, 10-14, 10-24
V_{RL} 10-1, 10-3, 10-14, 10-24
V_{SRC} 10-23
V_{SRC} 10-23
V_{SS} 5-21
V_{SSA} 10-1, 10-3, 10-14
V_{SSSYN} 5-5, 5-21
V_{STBY} 6-2

–W–

W 5-9, 5-12, 5-14, 5-17, 5-19, D-8
WAIT 7-3, 8-3, D-29, D-82
Wait states field (WAIT) D-29, D-82
WAKE 11-22, D-49
Wakeup
    address mark (WAKE) 11-22, D-49
WCOL 11-12, D-53
Wired-OR
    mode
        bit (WOR) D-65
        for SCI pins (WOMS) 11-19, D-49
    open-drain outputs 11-11
WOMP 11-11
WOMS 11-19, D-49
WOR D-65
Word composition 4-7
Write
    collision
        bit (WCOL) 11-12, D-53
        occurence 11-12
    cycle 5-38
        flowchart 5-39

–X–

X 5-9, 5-12, 5-14, 5-17, 5-19, D-8
XFC 5-8, 5-16, 5-21
XK 4-3, 4-5
XTAL 5-6, 5-8, 5-21

–Y–

Y 5-9, 5-12, 5-14, 5-17, 5-19, D-8
YK 4-3, 4-5

–Z–

Z 4-4, D-3
Zero flag (Z) 4-4, D-3
ZK 4-3, 4-5